

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

магістра

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ

(назва теми)

ПІДТРИМКИ ПРОДАЖУ ЧОВНІВ

RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS
FOR SUPPORTING BOAT SALES

Виконав(ла): студент(ка) 2 курсу, групи КНТ-123м

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

КОЗІЙ Д.В.

(ПРІЗВИЩЕ та ініціали)

Керівник ЛЬОВКІН В.М.

(ПРІЗВИЩЕ та ініціали)

Рецензент ГОЛУБ Т.В.

(ПРІЗВИЩЕ та ініціали)

2024_____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет _____ КНТ _____
Кафедра _____ програмних засобів _____
Ступінь вищої освіти _____ магістр _____
Спеціальність _____ 121 Інженерія програмного забезпечення _____
(код і найменування)
Освітня програма (спеціалізація) _____ Інженерія програмного забезпечення _____
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф. _____
Сергій СУББОТІН
« _____ » _____ 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

КОЗІЯ Дмитра Володимировича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів підтримки продажу човнів. Research and Software Implementation of Methods for Supporting Boat Sales

керівник проєкту (роботи) к.т.н., доцент ЛЬОВКІН Валерій Миколайович _____,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від « 7 » жовтня 2024 року № 398

2. Строк подання студентом проєкту (роботи) 02 грудня 2024 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Проєктування програмного забезпечення. 4. Основні рішення щодо реалізації програмного забезпечення. 5. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	ЛЬОВКІН В.М., доцент		
Нормоконтроль	БЄЛОВА А.В., асистент		

7. Дата видачі завдання « 06 » вересня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2 тиждень	Розділ 1
3	Дослідження проблеми	3-5 тижні	Розділ 2
4	Проектування та розробка архітектури програми	6 тиждень	Розділ 3
5	Розробка програми	7-8 тижні	Розділ 4
6	Тестування та експериментальне дослідження	9-10 тижні	Розділ 5
7	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	11 тиждень	Додатки
8	Захист роботи	12 тиждень	

Студент(ка)

_____ Дмитро КОЗІЙ
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

_____ Валерій ЛЬОВКІН
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
92 с., 26 рис., 4 табл., 3 дод., 9 джерел.

ПРОДАЖ ЧОВНІВ, ДОБІРКА ЧОВНІВ, АНСАМБЛІ ДЕРЕВ РІШЕНЬ,
ПРОГНОЗУВАННЯ, МАШИННЕ НАВЧАННЯ.

Об'єкт роботи – процес продажу човнів. Предмет роботи – методи підтримки продажу човнів. Мета роботи – розробити програмне забезпечення підтримки продажу човнів для забезпечення рішень стосовно визначення вартості продажу на основі даних.

Матеріали, методи та технічні засоби: мова Python та Python-пакети для розробки, фреймворк веброзробки Django, система керування базами даних MySQL.

Результати. Створена програма призначена для підтримки виставлення власних нових або вживаних човнів для продажу шляхом створення відповідних оголошень, створення, підтвердження або відхилення заявок на придбання човнів, пошуку необхідних човнів. Користувачам надається можливість створювати власні добірки човнів, щоб повертатись до раніше відібраних.

Висновки. Наукова новизна роботи полягає в методі прогнозування вартості продажу човна, який на основі ознак оголошення про продаж човна визначає створення і навчання моделі як ансамблю дерев рішень Random Forest, що дозволяє провести рекомендаційне оцінювання вартості човна для формування підсумкового оголошення.

Галузь використання. Підтримка продажу човнів, включаючи нові та ті, що були у використанні, від індивідуальних власників або дилерів чи власників з невеликою кількістю човнів.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 92 pages, 26 figures, 4 tables, 3 appendixes, 9 sources.

BOAT SALES, BOAT SET, DECISION TREES ENSEMBLE, FORECASTING, MACHINE LEARNING.

The object of work is the process of selling boats. The subject of the work is methods of supporting the sale of boats. The objective of the work is to develop boat sales support software to provide data-driven sales value decisions.

Materials, methods and technical means: Python language and Python packages for development, Django web development framework, MySQL database management system.

The results. The created program is designed to support the listing of own new or used boats for sale by creating relevant ads, creating, confirming or rejecting requests for the purchase of boats, searching for the required boats. Users are given the option to create their own sets of boats to return to the previously selected ones.

Conclusions. The scientific novelty of the work is in the method of predicting the cost of selling a boat, which, based on the features of an announcement about the sale of a boat, determines the creation and training of a model as an ensemble of Random Forest decision trees, which allows for a recommended assessment of the value of a boat to form a final announcement.

Field of use. Facilitating the sale of boats, including new and used ones, from individual owners or dealers or owners with a small number of boats.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз проблеми та постановка завдань дослідження	10
1.1 Огляд вебзастосунків підтримки продажу човнів	10
1.2 Порівняння аналогів розробки вебзастосунку підтримки продажу човнів	14
1.3 Висновки за розділом 1	15
2 Матеріали і методи.....	17
2.1 Задача прогнозування вартості продажу човна	17
2.2 Метод прогнозування вартості продажу човна.....	18
2.3 Висновки за розділом 2	21
3 Проєктування програмного забезпечення	22
3.1 Вибір мови та засобів розробки.....	22
3.2 Моделювання роботи з програмою за функціональними вимогами	23
3.3 Проєктування структури бази даних.....	25
3.4 Висновки за розділом 3	34
4 Основні рішення щодо реалізації програмного забезпечення	35
4.1 Порядок взаємодії користувача з програмою підтримки продажу човнів	35
4.2 Рішення з реалізації вебзастосунку	36
4.3 Висновки за розділом 4	48
5 Експлуатація, тестування та експериментальне дослідження програми	49
5.1 Опис програми.....	49
5.2 Експлуатація програми.....	49
5.3 Експериментальне дослідження	53
5.4 Висновки за розділом 5	56
Висновки	57
Перелік джерел посилання	58

Додаток А Технічне завдання	59
Додаток Б Текст програми	64
Додаток В Слайди презентації.....	85

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

MAE – Mean Absolute Error;

MSE – Mean Squared Error;

RMSE – Root Mean Squared Error.

ВСТУП

Човни не мають такого розповсюдження як автомобілі, проте це окрема сфера, яка заслуговує уваги, тим більше зважаючи на кількість водойм, які є в Україні. Такі човни можуть мати дуже різні типи, розміри. Відповідно дуже різних користувачів з дуже різними потребами потрібно задовольнити відповідними програмними засобами, враховуючи особливості човнів порівняно з автомобілями, надаючи гнучкий пошук для цього.

Важливим кроком під час виставлення власного човна для продажу є визначення умов такого продажу, з яких головною є встановлення вартості човна. Зважаючи на специфіку ринку та потребу в нових продавцях човнів, тобто розширенні ринку, необхідно в обов'язковому порядку такий програмний засіб забезпечити рекомендаційною функцією, яка б могла запропонувати продавцю адекватну вартість для його човна. Саме на цьому і має бути зроблено акцент у виконуваний роботі. Це стосується не тільки програмної реалізації відповідних можливостей, але і створення відповідного математичного інструменту, що має за допомогою використання адекватних, практично придатних засобів машинного навчання вирішити проблему прогнозування вартості продажу човнів.

Метою роботи є розробка програмного забезпечення підтримки продажу човнів для забезпечення рішень стосовно визначення вартості продажу на основі даних.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Огляд вебзастосунків підтримки продажу човнів

Продаж човнів підтримується різними програмними рішеннями, представленими спочатку вебзастосунками, а потім вже додатково можливо і застосунками інших типів.

Прикладом такого вебзастосунку підтримки продажу човнів є Boatsales.com.au (рис. 1.1) [1].

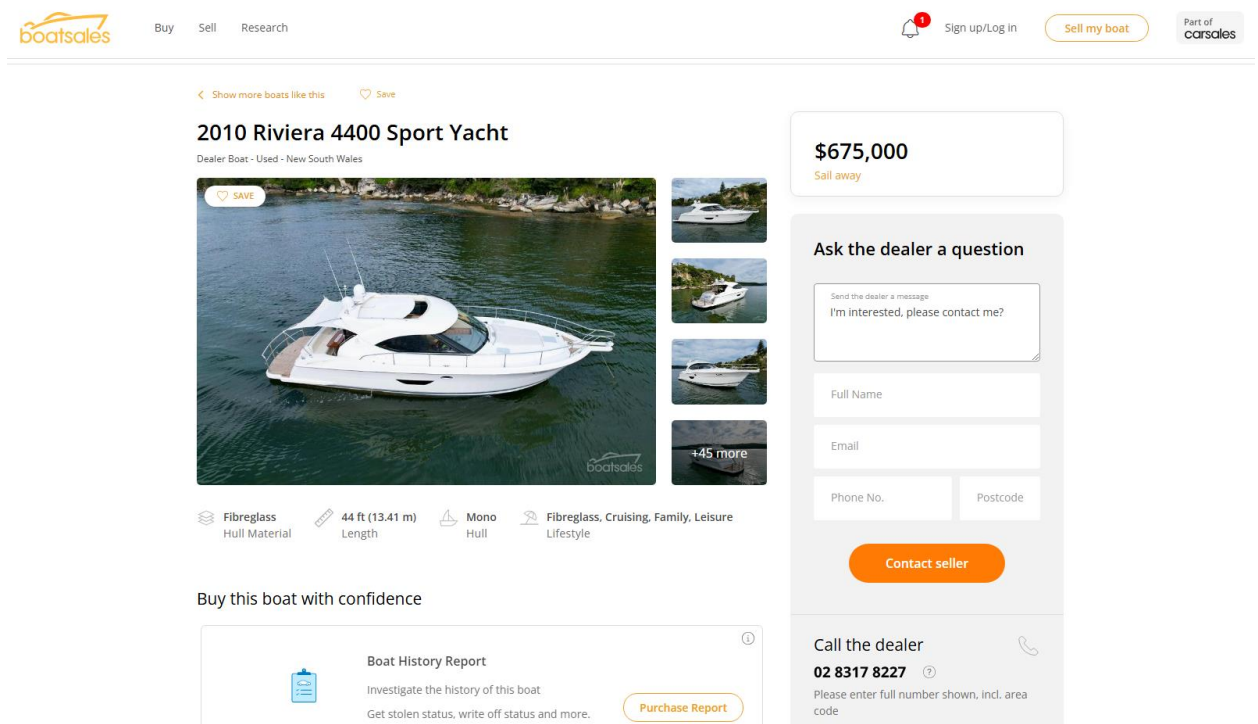
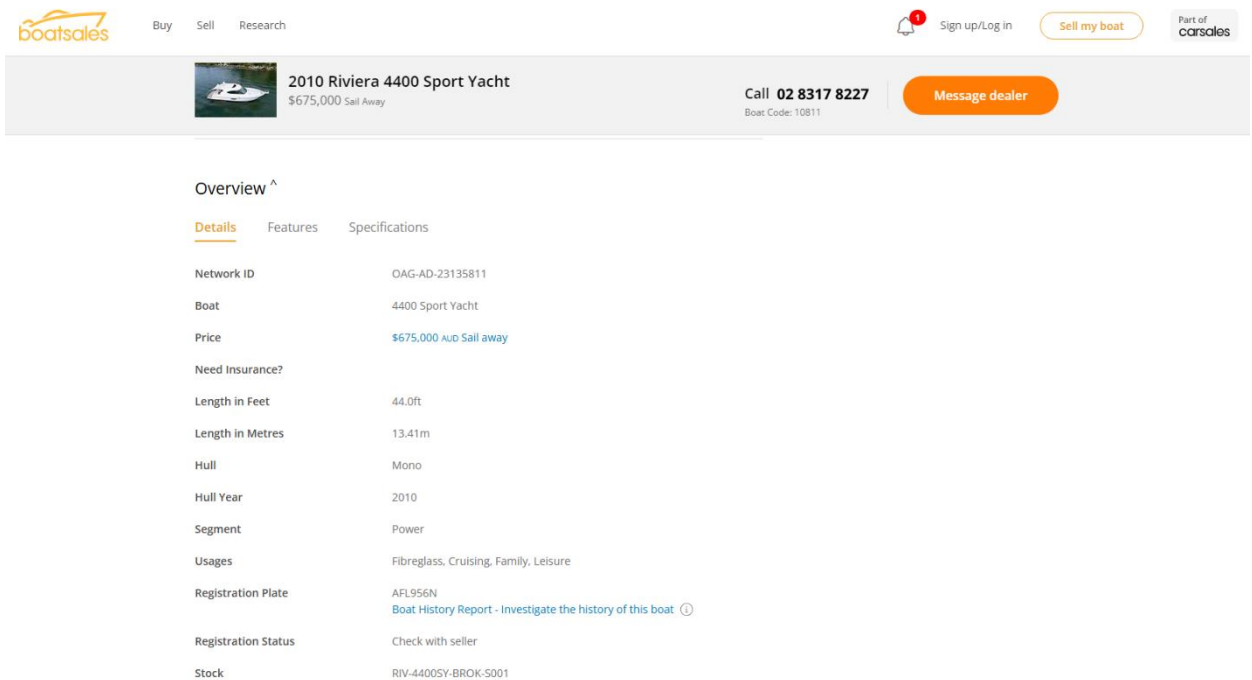


Рисунок 1.1 – Сторінка човна у програмі Boatsales.com.au

Boatsales.com.au дозволяє шукати човни, використовуючи різні фільтри, включаючи тип човна, виробника, ціль використання човна, ключове слово, обмеження на те, чи це тільки нові, чи тільки використовувані раніше човни, чи всі разом, місце розташування човнів, мінімальна та максимальна вартість човна.

За кожним човном доступний цілий ряд характеристик, які дозволяють визначитися з тим, на скільки він підходить під інтереси клієнта (рис. 1.2).



boatsales Buy Sell Research Sign up/Log in Sell my boat Part of carsales

2010 Riviera 4400 Sport Yacht
\$675,000 *Sail Away*

Call 02 8317 8227
Boat Code: 10811

Message dealer

Overview ^

Details Features Specifications

Network ID	OAG-AD-23135811
Boat	4400 Sport Yacht
Price	\$675,000 <i>Ask Sail away</i>
Need Insurance?	
Length in Feet	44.0ft
Length in Metres	13.41m
Hull	Mono
Hull Year	2010
Segment	Power
Usages	Fibreglass, Cruising, Family, Leisure
Registration Plate	AFL956N Boat History Report - Investigate the history of this boat
Registration Status	Check with seller
Stock	RIV-4400SY-BROK-5001

Рисунок 1.2 – Характеристики човна у програмі Boatsales.com.au

Клієнт може ставити запитання продавцю човна, спілкуватися таким чином з ним. Човни можна зберігати для того, щоб потім додатково переглядати. Будь-який користувач може виставляти свій човен на продаж, а далі і керувати таким оголошенням про продаж човна.

AutoRia (рис. 1.3) є програмним рішенням, яке відоме щодо продажу автомобілів, нерухомості, але також надає можливості продажу і водного транспорту, тобто човнів [2]. Проте варто вказати, що в даному випадку мова йде про те, що такі можливості є додатковими до роботи з автомобілями та нерухомістю. Зокрема сам пошук є спільним для різних напрямів, відповідно необхідно обирати з переліку водний транспорт. При цьому в самих фільтрах пошуку доступні тип кузова транспорту, країна виробник, марка човна, модель човна, рік виробництва як інтервал можливих років та вартість як інтервал можливих значень.

Проте порівняно з Boatsales.com.au AutoRia має також і дещо обмежений ряд характеристик самого човна. Звісно, що це можна деталізувати в описі, однак таким чином втрачається структуризація цієї інформації, можливість шукати за нею, що не зовсім є зручним. Відповідно щодо AutoRia

можна вказати на обмеженість застосування для саме продажу човнів (рис. 1.4).

The screenshot shows the AutoRia website's search interface. At the top, there's a navigation bar with links to RIA.com, Автомобілі, Недрухомість, Автозапчастини, and Збір на авто для ЗСУ. A green button '+ Продати авто' is on the right. Below the navigation bar, there's a banner with the text 'Шукаєте авто? ЗСУ теж потребують – допоможи їм, а ми подвоїмо'. The main section is titled 'Розширений пошук вживаних авто'. It features a 'New!' label and a 'Всі авто' button. There are checkboxes for 'Перевірений VIN-код', 'Держ. номер', 'Вперше на AUTO.RIA', and 'Електрокари'. The 'Тип транспорту' is set to 'Водний транспорт'. Under 'Тип кузова', there are checkboxes for 'Гідроцикл спортивний', 'Гідроцикл туристичний', 'Гідроцикли', 'Земснаряд', 'Катер', and 'Моторна яхта'. There's a link 'Інші типи кузова'. The 'Країна виробник' is set to 'Оберіть країну'. The 'Марка, модель, рік' section has dropdowns for 'Оберіть марку', 'Оберіть модель', 'Рік, від', and 'Рік, до', along with links '+ Ще марка' and 'Виключити авто'. The 'Вартість' section has input fields for 'Від', 'До', and a currency selector '\$'. At the bottom, it says 'Знайдено 346 авто' and has a blue 'Пошук' button.

Рисунок 1.3 – Сторінка пошуку водного транспорту в програмі AutoRia

Сам опис човна в програмі AutoRia включає фотографії, назву, текстовий опис, тип човна, кількість місць у човні, загальний пробіг, двигун та паливо двигуна, лакофарбове покриття, технічний стан, особливості салону та комфорту, наявні засоби мультимедіа, вартість, за яку власник може продати човен. Також можна звернути увагу, що навіть переключення між різними човнами організовано таким чином, що напис містить «наступне авто».

RIA.com Автономії Нерухом... Автозапчастини Збір на авто для ЗСУ Увійти в кабінет

auto RIA Вживані авто Нові авто Новини Все для авто + Продати авто

Знаємо, бо перевірили

AUTO.RIA.com Київська область Київ Azimut Flybridge-Range Azimut Flybridge-Range Київ

Повернутись до пошуку Наступне авто Crownline 220

Azimut Flybridge-Range 2011

294 300 \$ моніторинг цін
278 758 € • 12 351 771 грн [дивитись курс](#)
1 тис. км пробіг

Продавець
Володимир
• На сайті був сьогодні о 18:31

Київ

- Працює з AUTO.RIA більше 5 років
- Перевірений банком
- Перевірений телефон

(050) xxx-xx-xx [показати](#)

Про це авто ще немає коментарів від покупців

Знаєте більше про це авто?
Залиште коментар і ми перевіримо, щоб все було чесно

[Розказати правду](#)

201 412 Інтернет-інспекторів AUTO.RIA
9 274 коментарі за місяць
Як приєднатись до спільноти Інтернет-інспекторів?
Все про перевірки авто та продажів
Про безпечні купівлі та продажі

Погодження створене 12 грудня
ID авто 37333130
Переглядів авто 69
Наразі в Обраному 8
Опубліковано в ТОП 1

[Підняти оголошення в ТОП](#)

З цим авто також дивились

Azimut Flybridge-Range 38 2011
295 000 \$ • 1 тис. км

Azimut Flybridge-Range 2011

Моторна яхта • 12 місць
Пробіг 1 тис. км
Двигун 5.9 л (760 к.с. / 559 кВт) • Дизель

Опис
AZIMIT 38 FLYBRIDGE 40ft 2010 рік

Яхта в ідеальному стані, з самою максимальною комплектацією та сучасним дизайном, знаходиться в Києві
Перша реєстрація - 2011 рік
Розмитнена (прапор України)
[Читати ще](#)

Лакофарбове покриття
Як нове

Технічний стан
Повністю непошкоджене

Салон та комфорт
Люк • Бортовий комп'ютер • Декоративне підсвічування салону • Розетка 220V • Розетка 12V

Мультимедія
Акустика • Навігаційна система • AUX • Bluetooth • USB

[Залишити нотатку](#) [Перейти до порівняння](#)

Рисунок 1.4 – Сторінка човна в програмі AutoRia

Тож загалом в Україні такі програмні засоби дуже обмежені в своїх можливостях. Якщо додатково розглядати відповідні світові аналоги, то можна виділити Boat24 (рис. 1.5) [3]. Це рішення надає широкий набір фільтрів пошуку човнів, включаючи категорію, вартість, розміри, місце розташування, рік виробництва, тип пропозиції (включає зокрема вживаний човен, спільне використання човна, новий човен або новий човен на замовлення, а також

стан), виробника. Також доступний розширений пошук, який за човном дозволяє задати ще більше кількості параметрів.

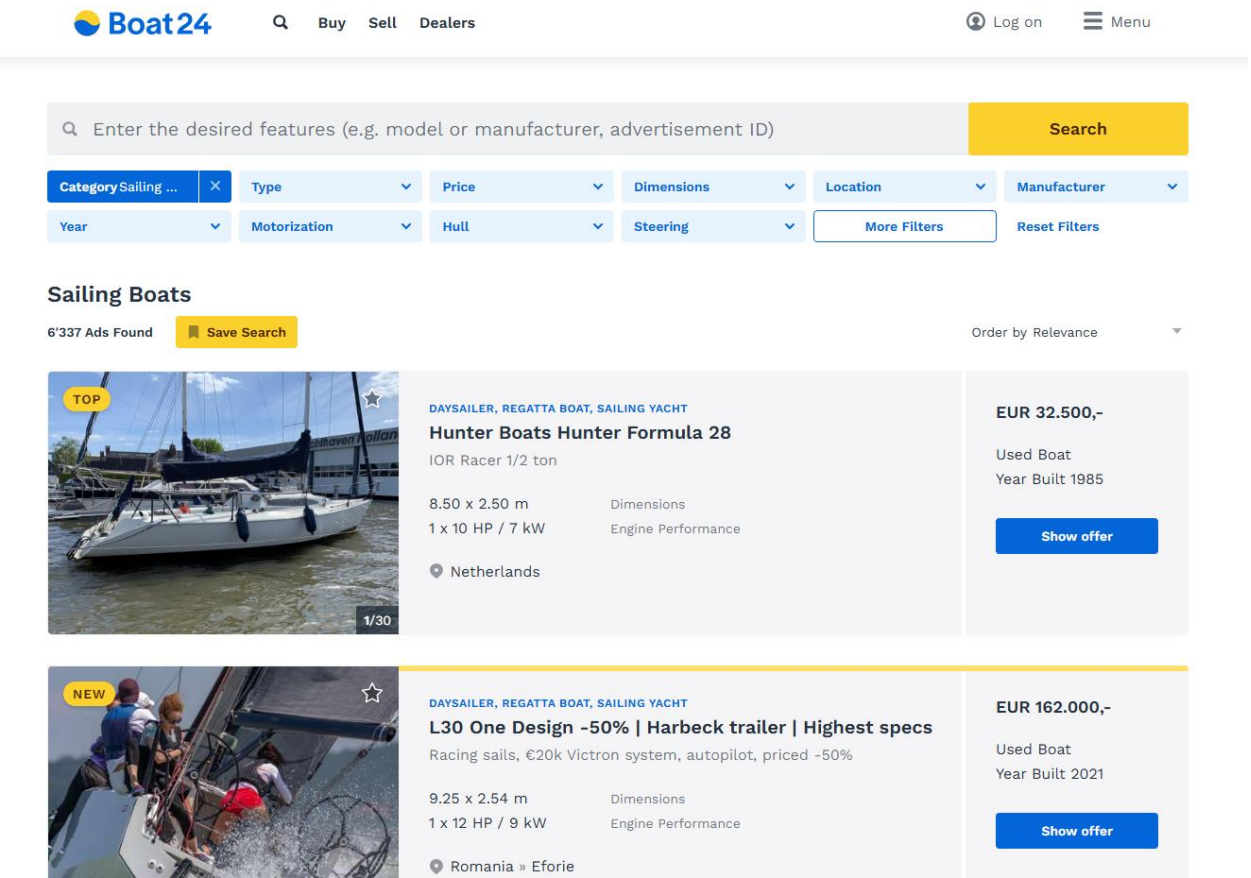


Рисунок 1.5 – Сторінка пошуку човнів у програмі Boat24

За човном також видається велика кількість характеристик з відповідними значеннями.

1.2 Порівняння аналогів розробки вебзастосунку підтримки продажу човнів

До фінального порівняння внесено окрім самої програми підтримки продажу човнів, яку потрібно створити, Boatsales.com.au, оскільки вона має достатньо близькі можливості з Boat24 зокрема, та AutoRia, що представляє стан галузі в Україні (табл. 1.1).

Таблиця 1.1 – Порівняння аналогів розробки вебзастосунку підтримки продажу човнів

Показник	Boatsales.com.au	AutoRia	Розробка
Багатопараметризований пошук човнів	Розширений	Стандартний	Розширений
Характеристики човна в оголошенні	Розширені	Стандартні	Розширені
Спеціалізація	Човни	Автомобілі, нерухомість, водний транспорт	Човни
Виставлення човна на продаж	+	+	+
Створення добірок човнів	—	—	+
Рекомендування вартості продажу човна для власника	—	—	+

Виключними особливостями розробки програми підтримки продажу човнів є наявність можливості створення добірок човнів, рекомендування вартості продажу човна для власника. Додатковою особливістю, яка є зручною для українського ринку є наявність розширених характеристик у човна в оголошенні та багатопараметризованого пошуку човнів.

1.3 Висновки за розділом 1

Завдання, які передбачаються для досягнення мети розробки програмного забезпечення підтримки продажу човнів для забезпечення рішень стосовно визначення вартості продажу на основі даних:

- виконати аналіз підтримки продажу човнів;
- розробити метод визначення рекомендованої вартості продажу човна;
- виконати проєктування програми підтримки продажу човнів;
- реалізувати програму підтримки продажу човнів;
- виконати експериментальне дослідження прогнозування вартості продажу човна.

У розділі 1 в результаті виконання огляду вебзастосунків підтримки продажу човнів, в якому в основному було розглянуто програми Boatsales.com.au, AutoRia, Boat24, враховано їх основні можливості та способи підтримки цих можливостей, визначено основні недоліки, а також як результат встановлено ті характеристики програми, які мають бути відмінностями програми, що розробляється. Виключними особливостями розробки програми підтримки продажу човнів є наявність можливості створення добірок човнів, рекомендації вартості продажу човна для власника.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Задача прогнозування вартості продажу човна

Продаж човнів обов'язково базується на формуванні оголошення. Частиною формування оголошення є визначення вартості човна, за якою має відбуватися продаж.

У деяких ситуаціях власник човна одразу має сталу думку щодо того, за якою вартістю має бути продано човен. Однак, в інших випадках таку вартість потрібно сформулювати. А для цього необхідне винесення рекомендації щодо човна.

У другому випадку винесення рекомендації щодо вартості човна реалізується на основі наявних даних, що представлені вибіркою даних про ті човни, які вже виставлені на продаж або раніше були виставлені.

Задача прогнозування вартості продажу човна полягає в тому, щоб на основі спостережень, які характеризують створені раніше оголошення про продаж човнів, визначити певного вигляду функціональну залежність між вхідними ознаками, що встановлюють характеристики човна, та вартістю човна.

Вхідні ознаки для моделі, яка і формує цю функціональну залежність, представлені наступним набором:

- тип човна (наприклад, спортивний човен, моторна яхта або інший подібний тип);
- виробник човна;
- додаткові характеристики типу човна (наприклад, човен, що був у використанні, або новий, дизельний човен тощо);
- рік виготовлення човна;
- довжина човна;
- ширина човна;
- основний матеріал, з якого виготовлений човен;
- локація, де човен розташований;

– кількість переглядів оголошення про продаж відповідного човна за останні 7 днів.

Вихідною ознакою є вартість човна.

Відповідно для розв’язання цієї задачі, яка за суттю машинного навчання є задачею прогнозування, необхідно створити модель, яка би на основі навчальної вибірки даних встановила залежність між вхідними ознаками, що характеризують оголошення про продаж човна та сам човен, та вартістю човна. При цьому набір вхідних ознак може бути зменшеним або перетвореним з початкового за необхідності, адже це може впливати на результати прогнозування.

Тоді для того, щоб таку задачу прогнозування вартості продажу човна розв’язати, необхідно створити відповідний метод. Метод прогнозування вартості продажу човна повинен визначати те, які вхідні ознаки мають стати ознаками моделі та в якому саме вигляді, на основі якої моделі прогнозування має виконуватись, яка структура навчальної вибірки даних має бути запропонована.

2.2 Метод прогнозування вартості продажу човна

Запропонований метод прогнозування вартості продажу човна, який на основі ознак оголошення про продаж човна визначає створення і навчання моделі як ансамблю дерев рішень Random Forest, що дозволяє провести рекомендаційне оцінювання вартості човна для формування підсумкового оголошення, формує наукову новизну роботи. Суть даного методу полягає в такій послідовності етапів.

На першому етапі формується навчальна вибірка даних про продаж човнів.

Для формування навчальної вибірки потрібно спершу перетворити вартість човнів у єдині одиниці. Грошові одиниці можуть вибиратися у загальній валюті або у місцевій валюті. Головне, щоб всі значення були

переведені до єдиної валюти, використовуючи адекватний курс обміну валют. Адекватним курсом обміну валют є той курс, який використовувався на момент створення оголошення про продаж відповідного човна. Ця логіка передбачає можливість застосування даного методу для роботи з оголошеннями про продаж човнів з різних країн та відповідно головне у різних валютах. Якщо використовується одна й та сама валюта для обчислень початково, то така проблема взагалі не виникає.

Для таких вхідних ознак має бути встановлено категоріальне значення, тобто кожне можливе значення відповідної ознаки має бути перетворено на відповідне число, яке це значення унікально ідентифікує (всі унікальні значення таким чином мають бути встановлені):

- тип човна;
- виробник човна;
- додаткові характеристики типу човна;
- основний матеріал, з якого виготовлений човен;
- локація, де човен розташований.

Окрім того з вибірки мають бути вилучені всі дублікати оголошень про продаж човнів.

Вартість човна має бути нормалізована для знаходження в інтервалі від 0 до 1. Для цього відповідно визначається мінімальна та максимальна існуючі вартості продажу човна в оголошеннях у навчальній вибірці даних.

На другому етапі має бути створено модель прогнозування вартості продажу човна, для чого запропоновано скористатись ансамблем дерев рішень на основі методу Random Forest [4]-[5], задаючи відповідну кількість дерев у ансамблі при цьому.

На третьому етапі має бути реалізовано за підготовленою вибіркою даних про продаж човнів навчання моделі на основі методу Random Forest. У процесі реалізації програми навчена модель прогнозування вартості продажу човнів має бути збережена та завантажуватись для виконання прогнозування вартості продажу човна, коли така необхідність виникає. Моментом

необхідності є момент створення або відповідно редагування оголошення про продаж човна.

На четвертому етапі створюється оголошення про продаж човна, а за цим оголошенням виконується формування характеристик човна як вхідних ознак моделі. Це фактично екземпляр даних з початкової вибірки, як вона формувалась на першому етапі, проте вже без встановленої вартості човна. Вартість човна обчислюється програмно за допомогою навченої моделі на попередньому, третьому етапі (рис. 2.1).

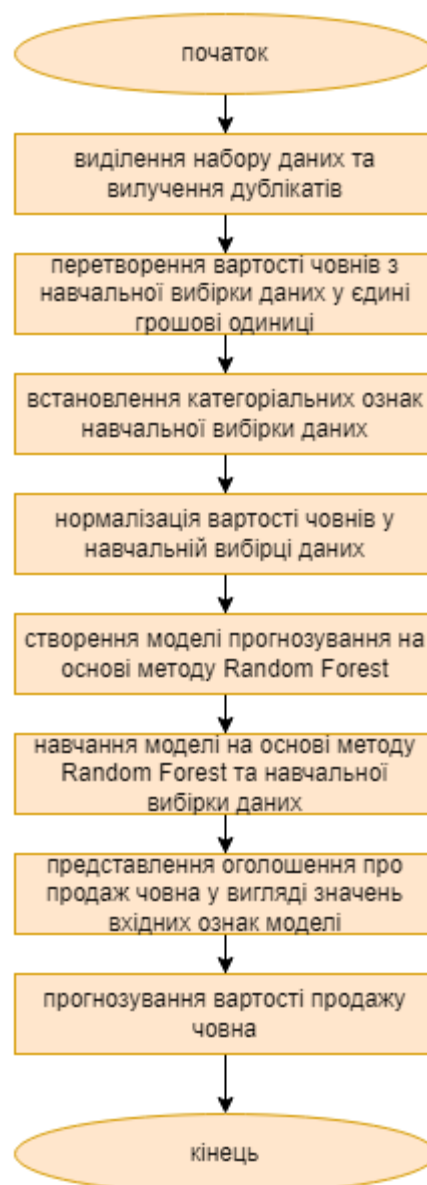


Рисунок 2.1 – Метод прогнозування вартості продажу човна

2.3 Висновки за розділом 2

У підсумку розділ 2 як основний результат має наукову новизну роботи, що полягає в методі прогнозування вартості продажу човна, який на основі ознак оголошення про продаж човна визначає створення і навчання моделі як ансамблю дерев рішень Random Forest, що дозволяє провести рекомендаційне оцінювання вартості човна для формування підсумкового оголошення.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір мови та засобів розробки

Порівняння під час вибору вебфреймворку для розробки програми підтримки продажу човнів зведено до вебфреймворків Django [6] та Laravel [7] (табл. 3.1).

Таблиця 3.1 – Вибір вебфреймворку для розробки програми підтримки продажу човнів

	Django	Laravel
Тип пропонованого рішення	Вебфреймворк	Вебфреймворк
Мова створення	Python	PHP
Підтримка високого трафіку	Придатний	Потребує додаткової оптимізації для забезпечення того ж рівня
Рівень підтримки машинного навчання	Високий, широка кількість актуальних засобів	Гірший
Безпека	Високий рівень підтримки, велика кількість практик	Високий рівень підтримки

Підсумково обрано Django, адже цей вебфреймворк має кращу підтримку високого трафіку, що дозволяє краще масштабувати проєкт в подальшому, окрім того дещо більшу кількість засобів підтримки безпеки та за рахунок мови Python кращий рівень підтримки машинного навчання.

3.2 Моделювання роботи з програмою за функціональними вимогами

У процесі моделювання роботи з програмою підтримки продажу човнів було визначено такий ряд сценаріїв для користувача:

- авторизація;
- пошук човнів для придбання;
- створення оголошення про продаж човна, включаючи визначення рекомендованої ціни продажу човна;
- відтворення переліку оголошень про продаж човнів у користувача;
- відтворення переліку оголошень про продаж власних човнів;
- перегляд оголошення про продаж човна;
- вподобання човна з оголошення;
- відтворення переліку човнів, уподобаних користувачем;
- створення добірки човнів;
- додавання човна до добірки;
- подання заявки на придбання човна;
- підтвердження заявки на придбання човна;
- відтворення переліку заявок на придбання човна;
- відтворення власних заявок на придбання човна;
- відтворення переліку човнів з добірки;
- відтворення переліку добірок користувача;
- адресування запитання продавцю човна;
- відправлення відповіді на запитання продавцю човна;
- відтворення переліку всіх покупців човнів, які поставили запитання;
- відтворення переліку всіх продавців, яким поставив запитання покупець;
- відтворення запитань між продавцем та покупцем човна;
- перегляд профілю користувача програми підтримки продажу човнів

(рис. 3.1).

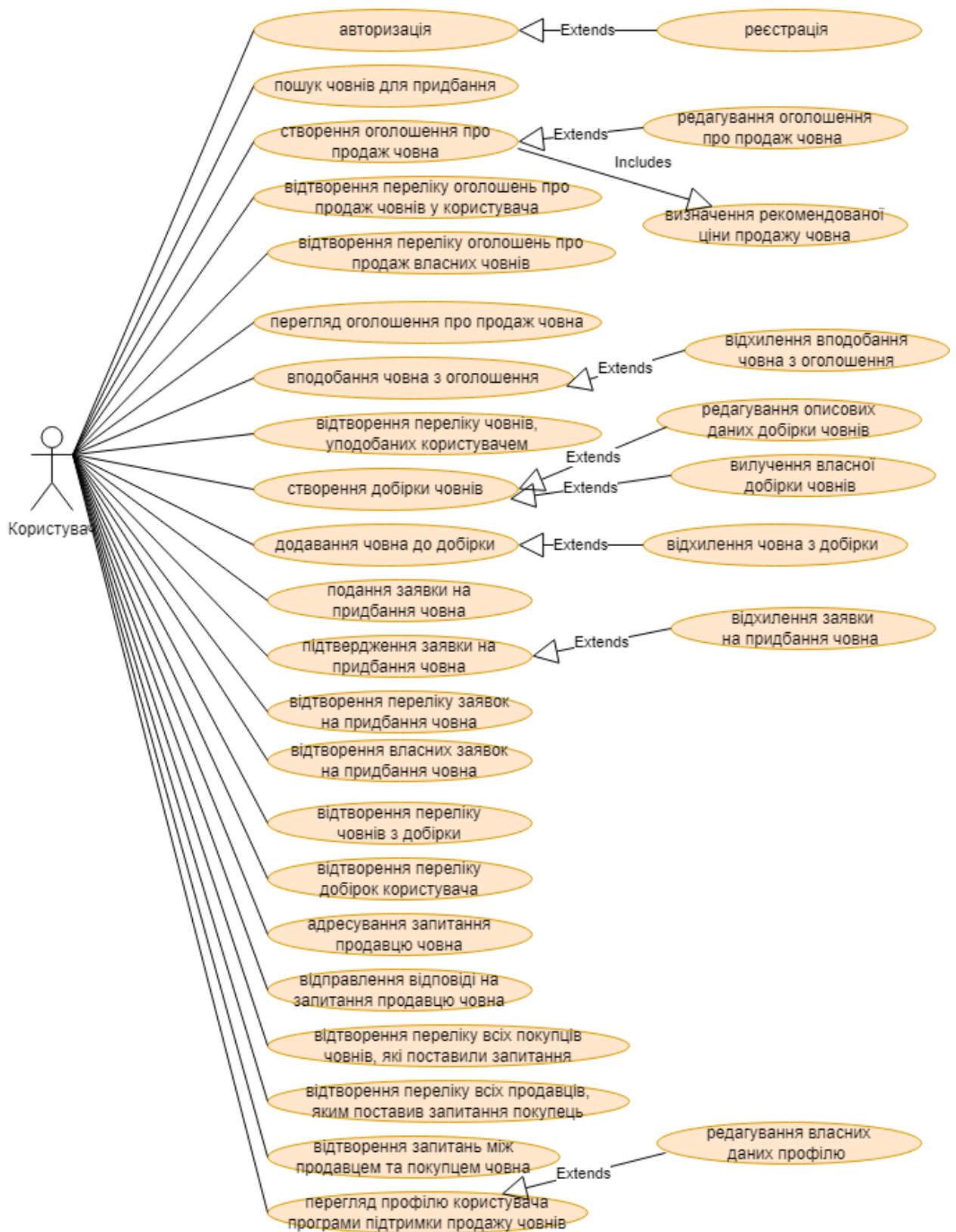


Рисунок 3.1 – Діаграма прецедентів користувача програми підтримки продажу човнів

Альтернативні варіанти:

- реєстрація є альтернативним для авторизації, але за умов, що обліковий запис у користувача відсутній;
- редагування оголошення про продаж човна є альтернативним для створення оголошення про продаж човна, якщо за цим човном у цього користувача вже наявне оголошення;
- відхилення вподобання човна з оголошення є альтернативним для вподобання човна з оголошення, коли вподобання човна цим користувачем присутнє;
- редагування описових даних добірки човнів є альтернативним для створення добірки човнів, коли ця добірка човнів вже є у цього користувача;
- відхилення човна з добірки є альтернативним для додавання човна до добірки, коли цей човен до цієї добірки доданий;
- відхилення заявки на придбання човна є альтернативним для підтвердження заявки на придбання човна, якщо власник човна та відповідного оголошення про продаж човна не бажає продавати його відповідній особі;
- редагування власних даних профілю є альтернативним для перегляду профілю користувача програми підтримки продажу човнів;
- вилучення власної добірки човнів є альтернативним для створення добірки човнів, коли добірка човнів створена цим користувачем, але вже не потрібна.

3.3 Проєктування структури бази даних

MySQL є системою керування, що забезпечує роботу з даними у вигляді підтримки відповідної бази даних, яку структуровано нижче. Ця система є широковідомою, а тому має високий рівень підтримки, робота з Django її достатньо потужно інтегрована, а сам інструмент не вимагає сплати за використання.

Таблиця Userprof необхідна в структурі бази даних для збереження даних профілю користувача, який може бути і клієнтом, і продавцем човнів, відповідно організована ця таблиця за допомогою вже наявних даних про користувача Django в таблиці user, тому набір полів включає:

- id – ідентифікатор профілю користувача застосунку підтримки продажу човнів;
- user – обліковий запис користувача в Django, за яким працює цей користувач застосунку підтримки продажу човнів;
- surandname – повні ім'я та прізвище профілю користувача застосунку підтримки продажу човнів;
- pic – профільна фотографія користувача застосунку підтримки продажу човнів.

Таблиця Boattype необхідна в структурі бази даних для збереження даних типів човнів, які можуть їм присвоюватися в конкретних оголошеннях, включаючи поля:

- id – ідентифікатор типу човна;
- boattypetitle – назва типу човна.

Типами човнів можуть бути, наприклад, моторна яхта, спортивний човен, катамаран тощо.

Таблиця Boatmanufacturer необхідна в структурі бази даних для збереження даних компаній-виробників човнів, включаючи поля:

- id – ідентифікатор компанії-виробника човна;
- manufacturertitle – назва компанії-виробника човна;
- webaddress – адреса вебсайту компанії-виробника човна.

Таблиця Locationregion необхідна в структурі бази даних для збереження даних регіонів, за якими укрупнюється розташування міст, де можуть знаходитись човни (в залежності від розташування і направленості вебзастосунку це можуть бути або країни, або регіони чи області в конкретній країні, якщо продажу човнів реалізується для однієї країни), включаючи поля:

– id – ідентифікатор регіону, за яким укрупнюється розташування міст, де можуть знаходитись човни;

– regiontitle – назва регіону, за яким укрупнюється розташування міст, де можуть знаходитись човни.

Таблиця Location необхідна в структурі бази даних для збереження даних міст, де можуть знаходитись човни під час продажу, включаючи поля:

– id – ідентифікатор міста, де можуть знаходитись човни під час продажу;

– locationregion – регіон, в якому це місто, де може знаходитись човен на продажу, знаходиться, відповідно один-до-багатьох є типом зв'язку цієї таблиці з таблицею Locationregion за цим ключем як зовнішнім;

– centertitle – назва міста, де можуть знаходитись човни під час продажу.

Таблиця Boatmodel необхідна в структурі бази даних для збереження даних таблиць моделей човнів, включаючи поля:

– id – ідентифікатор моделі човна:

– boatmanufacturer – компанія-виробник човнів, що виготовляє цю модель човна, відповідно один-до-багатьох є типом зв'язку цієї таблиці з таблицею Boatmanufacturer за цим ключем як зовнішнім;

– boattype – тип човна, який представляє ця модель човна;

– modeltitle – сама назва моделі човна.

Таблиця Boat необхідна в структурі бази даних для збереження даних оголошень про продаж човнів, включаючи поля:

– id – ідентифікатор оголошення, за яким продається човен;

– dealerprof – профіль власника човна, яким він представлений як користувач застосунку підтримки продажу човнів, відповідно один-до-багатьох є типом зв'язку цієї таблиці з таблицею Userprof за цим ключем як зовнішнім;

– adtitle – назва оголошення про продаж човну (може використовуватися як розширений варіант моделі човна, але при цьому

враховуючи обмеження на відображення назви в результатах пошуку човнів та будь-яких подібних елементах);

- location – місто, де знаходиться човен, який в цьому оголошенні пропонується для придбання, відповідно один-до-багатьох є типом зв'язку цієї таблиці з таблицею Location за цим ключем як зовнішнім;

- boatmodel – модель човна, відповідно один-до-багатьох є типом зв'язку цієї таблиці з таблицею Boatmodel за цим ключем як зовнішнім;

- builtyear – рік виготовлення човна, який в оголошенні пропонується для придбання;

- regplate – реєстраційний номер човна, який в оголошенні пропонується для придбання;

- width – ширина корпусу човна, який в оголошенні пропонується для придбання;

- length – довжина корпусу човна, який в оголошенні пропонується для придбання;

- conditiontype – стан човна, який в оголошенні пропонується для придбання;

- boatcomment – опис для оголошення човна, який в оголошенні пропонується для придбання;

- price – вартість придбання човна, який пропонується в оголошенні;

- pricerecom – рекомендована вартість продажу човна за запропонованим методом прогнозування вартості продажу човна;

- hulltype – тип корпусу човна, який в оголошенні пропонується для придбання;

- watercapacity – водоемність човна, який в оголошенні пропонується для придбання;

- hullmaterial – матеріал, з якого виготовлений корпус човна, який в оголошенні пропонується для придбання, відповідно один-до-багатьох є типом зв'язку цієї таблиці з таблицею Buildmaterial за цим ключем як зовнішнім;

– deckmaterial – матеріал, з якого виготовлена палуба човна, який в оголошенні пропонується для придбання, відповідно один-до-багатьох є типом зв'язку цієї таблиці з таблицею Buildmaterial за цим ключем як зовнішнім;

– showernumber – кількість душових кабін на човні, який в оголошенні пропонується для придбання;

– toiletnumber – кількість туалетів на човні, який в оголошенні пропонується для придбання;

– anchorwinch – текстовий опис особливостей якірної лебідки човна, який в оголошенні пропонується для придбання;

– bilgerump – опис особливостей трюмного насосу човна, який в оголошенні пропонується для придбання;

– covers – особливості покриття човна, який в оголошенні пропонується для придбання;

– entertainment – система для розваг (музична система) човна, який в оголошенні пропонується для придбання;

– epirb – служба радіозв'язку човна, який в оголошенні пропонується для придбання, що застосовується під час надзвичайних ситуацій;

– lifejackets – кількість рятувальних жилетів на човні, який в оголошенні пропонується для придбання;

– navigation – навігаційна система човна, який в оголошенні пропонується для придбання;

– radiocommunication – система радіокомунікації човна, який в оголошенні пропонується для придбання;

– enginefuelcapacity – обсяг палива, який здатен вмістити човен, який в оголошенні пропонується для придбання, що використовується для роботи двигуна;

– enginehours – кількість годин, яку за повного баку в середньому може працювати човен, який в оголошенні пропонується для придбання;

- enginehorsepowers – кількість кінських сил двигуна човна, який в оголошенні пропонується для придбання;
- enginereleaseyear – рік виготовлення поточного встановленого двигуна на човні, який в оголошенні пропонується для придбання;
- displacement – кількість виконаних замін двигуна човна, який в оголошенні пропонується для придбання;
- enginemade – основний матеріал, який використовується при роботі двигуна човна, який в оголошенні пропонується для придбання;
- enginefueltype – тип палива, необхідний для роботи двигуна човна, який в оголошенні пропонується для придбання;
- state – стан оголошення про продаж човна;
- creationdate – дата створення оголошення про продаж човна.

Таблиця Userprofboatset необхідна в структурі бази даних для збереження даних добірок човнів, включаючи поля:

- id – ідентифікатор користувацької добірки човнів;
- userprof – профіль користувача застосунку підтримки продажу човнів, який створив цю добірку, відповідно один-до-багатьох є типом зв'язку цієї таблиці з таблицею Userprof за цим ключем як зовнішнім;
- setname – назва користувацької добірки човнів.

Таблиця Userprofboatsetboat необхідна в структурі бази даних для збереження даних човнів з добірок, включаючи поля:

- id – ідентифікатор входження човна до користувацької добірки;
- userprofboatset – користувацька добірка човнів, у яку входить цей човен, відповідно один-до-багатьох є типом зв'язку цієї таблиці з таблицею Userprofboatset за цим ключем як зовнішнім;
- boat – човен, який входить у цю користувацьку добірку, відповідно один-до-багатьох є типом зв'язку цієї таблиці з таблицею Boat за цим ключем як зовнішнім.

Таблиця Boatpic необхідна в структурі бази даних для збереження даних фотографій човнів, що знаходяться у продажі, включаючи поля:

- id – ідентифікатор фотографії човна;
- boat – човен, який показаний на фотографії, відповідно один-до-багатьох є типом зв'язку цієї таблиці з таблицею Boat за цим ключем як зовнішнім;
- feature – поле, яке позначає, це основна фотографія для представлення оголошення про продаж човна чи це додаткова фотографія;
- pic – фотографія човна.

Таблиця Boatlike необхідна в структурі бази даних для збереження даних висловлених вподобань човнам, включаючи поля:

- id – ідентифікатор висловленого вподобання човну користувачем застосунку підтримки продажу човнів;
- boat – човен, якому користувач застосунку підтримки продажу човнів висловив уподобання, відповідно один-до-багатьох є типом зв'язку цієї таблиці з таблицею Boat за цим ключем як зовнішнім;
- userprof – користувач застосунку підтримки продажу човнів, який висловив уподобання човну, відповідно один-до-багатьох є типом зв'язку цієї таблиці з таблицею Userprof за цим ключем як зовнішнім.

Таблиця Boatview необхідна в структурі бази даних для збереження даних переглядів оголошень про продаж човнів користувачами застосунку підтримки продажу човнів, включаючи поля:

- id – ідентифікатор перегляду оголошення про продаж човнів користувачем застосунку підтримки продажу човнів;
- boat – човен, оголошення про продаж якого переглянув користувач застосунку підтримки продажу човнів, відповідно один-до-багатьох є типом зв'язку цієї таблиці з таблицею Boat за цим ключем як зовнішнім;
- userprof – користувач застосунку підтримки продажу човнів, який переглянув оголошення про продаж човну, відповідно один-до-багатьох є типом зв'язку цієї таблиці з таблицею Userprof за цим ключем як зовнішнім.

Таблиця Boatmaterial необхідна в структурі бази даних для збереження даних матеріалів, з яких виготовлений човен, включаючи поля:

- id – ідентифікатор матеріалу, з якого виготовлений човен;
- boat – човен, при виготовленні якого використаний цей матеріал, відповідно один-до-багатьох є типом зв'язку цієї таблиці з таблицею Boat за цим ключем як зовнішнім;
- buildmaterial – матеріал, використаний при виробництві човна, відповідно один-до-багатьох є типом зв'язку цієї таблиці з таблицею Buildmaterial за цим ключем як зовнішнім.

Таблиця Buildmaterial необхідна в структурі бази даних для збереження даних матеріалів, з яких можуть виготовлятися човни, включаючи поля:

- id – ідентифікатор матеріалу, з якого можуть виготовлятися човни;
- biuldmaterialtitle – назва матеріалу, з якого можуть виготовлятися човни.

Таблиця Boatsale необхідна в структурі бази даних для збереження даних заявок на придбання човнів, включаючи поля:

- id – ідентифікатор заявки на придбання човну, створеної користувачем застосунку підтримки продажу човнів;
- boat – човен, заявку на придбання якого залишив користувач застосунку підтримки продажу човнів, відповідно один-до-багатьох є типом зв'язку цієї таблиці з таблицею Boat за цим ключем як зовнішнім;
- userprof – користувач застосунку підтримки продажу човнів, який створив цю заявку на його придбання, відповідно один-до-багатьох є типом зв'язку цієї таблиці з таблицею Userprof за цим ключем як зовнішнім;
- apllicationdate – дата, час подання заявки на придбання човна, створеної користувачем застосунку підтримки продажу човнів;
- comment – коментар до заявки на придбання човна, створеної користувачем застосунку підтримки продажу човнів;
- state – стан заявки на придбання човна, створеної користувачем застосунку підтримки продажу човнів (рис. 3.2).

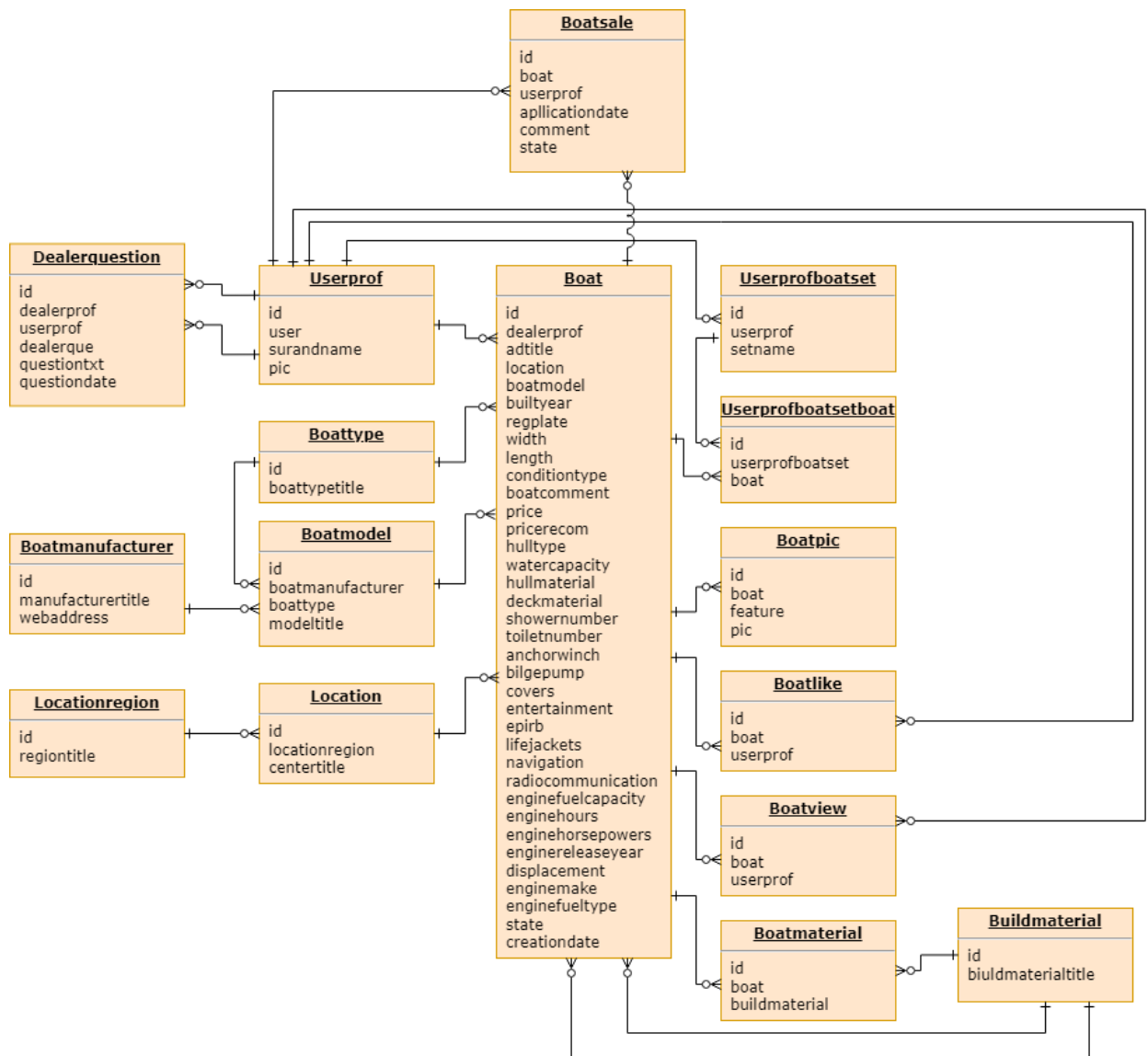


Рисунок 3.2 – Схема бази даних

Таблиця Dealerquestion необхідна в структурі бази даних для збереження даних запитань і відповідей між покупцями і продавцями човнів, включаючи поля:

- id – ідентифікатор запитання або відповіді між покупцем і продавцем човнів;
- dealerprof – продавець човна, якому направлено запитання або який створив відповідь на запитання покупця човна, відповідно один-до-багатьох є типом зв'язку цієї таблиці з таблицею Userprof за цим ключем як зовнішнім;

- userprof – покупець човна, якому направлено відповідь на запитання або який створив запитання до покупця човна, відповідно один-до-багатьох є типом зв'язку цієї таблиці з таблицею Userprof за цим ключем як зовнішнім;
- dealerque – поле, що визначає, це запитання покупця човна до його продавця чи це відповідь продавця човна на запитання покупця;
- questiontxt – текст запитання або відповіді між покупцем і продавцем човнів;
- questiondate – дата, час створення запитання або відповіді між покупцем і продавцем човнів.

3.4 Висновки за розділом 3

Розділ 3 стосується результатів, які подають виконаний процес проєктування програми підтримки продажу човнів, включаючи до результатів прецеденти використання програми клієнтами та продавцями човнів, вибір Python, Django і MySQL як відповідно мови, вебфреймворку розробки та системи керування базами даних, структуру якої деталізовано.

4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Порядок взаємодії користувача з програмою підтримки продажу човнів

Порядок роботи з програмою підтримки продажу човнів визначається діями користувача та охоплює ряд складових, доступ до яких визначається і вибирається користувачем (рис. 4.1).

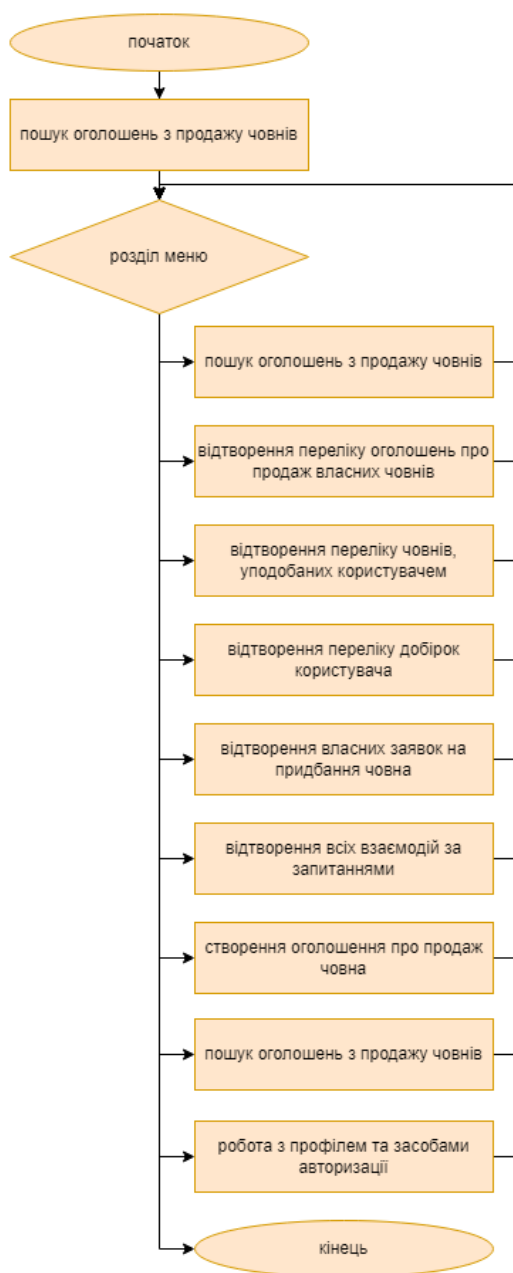


Рисунок 4.1 – Схема функціонування програми підтримки продажу човнів

4.2 Рішення з реалізації вебзастосунку

Представлення відтворення переліку оголошень про продаж власних човнів має реалізацію функцією `own_userprof_boats_view`. Параметр `sortorder` встановлює упорядкування оголошень про продаж власних човнів у переліку: 0 – за датою створення від нових, 1 – за датою створення від старих, 2 – за збільшенням кількості переглядів, 3 – за зменшенням кількості переглядів, 4 – за збільшенням кількості вподобань, 5 – за зменшенням кількості вподобань, 6 – за назвою за алфавітом, 7 – за назвою у протилежному напрямку від алфавіту. За оголошеннями про продаж власних човнів підраховується кількість переглядів та вподобань. За кожним оголошенням про продаж власних човнів визначається, чи його вподобав користувач, якого авторизовано, чи ні, а також чи він переглядав оголошення. Також виділяються добірки човнів та за ними визначається, чи входить в них човен з оголошення. Результати повертаються в шаблон вебсторінки `boats.html`, де за ключем `boats` містяться самі оголошення, а за ключем `boatsets` – добірки човнів. Реалізацію шаблонів сторінок вебзастосунку підтримки продажу човнів виконано за шаблоном [8], який є вільним.

Перегляд оголошення про продаж човна має реалізацію функцією `boat_detail_view`, яка має параметр `boatId`, що встановлює номер човна. За цим оголошенням про продаж човна визначається, чи його вподобав користувач, якого авторизовано, чи ні, а також чи він переглядав оголошення. Результати повертаються в шаблон вебсторінки `boat.html`, де за ключем `boat` міститься саме оголошення, а за ключем `boatsets` – добірки човнів.

Відтворення переліку човнів, уподобаних користувачем, має реалізацію функцією `liked_boats_view`. Вибираються в шаблон сторінки `lboats.html` ті човни, які були вподобані користувачем, який авторизувався. Параметр `sortorder` встановлює упорядкування оголошень про продаж власних човнів у переліку: 0 – за датою створення від нових, 1 – за датою створення від старих, 2 – за збільшенням кількості переглядів, 3 – за зменшенням кількості

переглядів, 4 – за збільшенням кількості вподобань, 5 – за зменшенням кількості вподобань, 6 – за назвою за алфавітом, 7 – за назвою у протилежному напрямку від алфавіту.

Представлення відтворення переліку оголошень про продаж човнів у користувача має реалізацію функцією `userprof_boats_view`. Параметр `userprofName` задає логін користувача. Параметр `sortorder` встановлює упорядкування оголошень про продаж човнів у переліку: 0 – за датою створення від нових, 1 – за датою створення від старих, 2 – за збільшенням кількості переглядів, 3 – за зменшенням кількості переглядів, 4 – за збільшенням кількості вподобань, 5 – за зменшенням кількості вподобань, 6 – за назвою за алфавітом, 7 – за назвою у протилежному напрямку від алфавіту. За оголошеннями про продаж човнів підраховується кількість переглядів та вподобань. За кожним оголошенням про продаж власних човнів визначається, чи його вподобав користувач, якого авторизовано, чи ні, а також чи він переглядав оголошення. Також виділяються добірки човнів та за ними визначається, чи входить в них човен з оголошення. Результати повертаються в шаблон вебсторінки `usboats.html`, де за ключем `boats` містяться самі оголошення, а за ключем `boatsets` – добірки човнів.

Відтворення історії перегляду оголошень про продаж човнів має реалізацію функцією `history_boats_view`. Параметр `sortorder` встановлює упорядкування оголошень про продаж власних човнів у переліку: 0 – за датою створення від нових, 1 – за датою створення від старих, 2 – за збільшенням кількості переглядів, 3 – за зменшенням кількості переглядів, 4 – за збільшенням кількості вподобань, 5 – за зменшенням кількості вподобань, 6 – за назвою за алфавітом, 7 – за назвою у протилежному напрямку від алфавіту. Результати повертаються в шаблон вебсторінки `hboats.html`, де за ключем `boats` містяться самі оголошення, а за ключем `boatsets` – добірки човнів.

Відтворення переліку човнів з добірки має реалізацію функцією `boatset_boats_view`. Параметр `boatsetId` задає номер добірки човнів. Параметр `sortorder` встановлює упорядкування оголошень про продаж човнів у переліку:

0 – за датою створення від нових, 1 – за датою створення від старих, 2 – за збільшенням кількості переглядів, 3 – за зменшенням кількості переглядів, 4 – за збільшенням кількості вподобань, 5 – за зменшенням кількості вподобань, 6 – за назвою за алфавітом, 7 – за назвою у протилежному напрямку від алфавіту. Результати повертаються в шаблон вебсторінки `boatset.html`, де за ключем `boats` містяться самі оголошення, а за ключем `boatsets` – існуючі добірки човнів.

Пошук човнів для придбання має реалізацію функцією `search_boats_view`, що результати повертає в шаблон вебсторінки `searchboats.html`. Параметри пошуку можуть містити виробника човнів (`manuf` з GET-запиту), модель човна (`model`), місце розташування човна (`location`), матеріал, з якого виготовлений корпус човна (`hullmaterial`), матеріал, з якого виготовлена палуба човна (`deckmaterial`), кількість душових кабін на човні в інтервалі від `startshowers` до `endshowers`, кількість туалетів на човні в інтервалі від `starttoilets` до `endtoilets`, кількість рятувальних жилетів в інтервалі від `startlifejackets` до `endlifejackets`, систему навігації (`navigation`), радіозв'язок (`radio`), назву оголошення про продаж човна або частину (`adtitle`), рік будівництва човна в інтервалі від `startbuiltyear` до `endbuiltyear`, ширина човна в інтервалі від `startboatwidth` до `endboatwidth`, довжина човна в інтервалі від `startboatlength` до `endboatlength`, стан човна (`condtype`), вартість човна в інтервалі від `startprice` до `endprice`, тип корпусу човна (`hulltype`), відоємність човна в інтервалі від `startcapacity` до `endcapacity`, обсяг палива, який здатен вмістити човен в інтервалі від `startenginefuelcapac` до `endenginefuelcapac`, кількість годин, яку здатен відпрацювати човен, в інтервалі від `startenginehours` до `endenginehours`, кількість кінських сил човна в інтервалі від `starthorsepowers` до `endhorsepowers`, рік виробництва човна в інтервалі від `starreleaseyear` до `endreleaseyear`, максимальна кількість виконаних замін двигуна `displacement`, основний матеріал, який використовується при роботі двигуна човна, `enginemade`, тип палива, необхідний для роботи двигуна човна, `enginefueltype`. Параметр `sortorder` встановлює упорядкування оголошень про продаж човнів

у переліку: 0 – за датою створення від нових, 1 – за датою створення від старих, 2 – за збільшенням кількості переглядів, 3 – за зменшенням кількості переглядів, 4 – за збільшенням кількості вподобань, 5 – за зменшенням кількості вподобань, 6 – за назвою за алфавітом, 7 – за назвою у протилежному напрямку від алфавіту. Результати повертаються в шаблон вебсторінки `searchboats.html`, де за ключем `boats` містяться самі оголошення, а за ключем `boatsets` – існуючі добірки човнів, за ключем `boattypes` – можливі типи човнів, `boatmaterials` – можливі матеріали при виготовленні човнів.

Створення оголошення про продаж човна з визначенням рекомендованої ціни продажу човна має реалізацію функцією `boat_ad_publish_view`, яка використовує клас форми `BoatForm` та звертається до функції `getpricekozumodel`, яка за даними оголошення про продаж човна виконує прогнозування його вартості на основі запропонованого методу. Редагування оголошення про продаж човна має реалізацію функцією `boat_ad_change_view`, яка має параметр `boatId`, де задається номер човна.

Створення добірки човнів має реалізацію функцією `userprofboatset_add_view`, яка використовує клас форми `UserprofboatsetForm`.

Редагування описових даних добірки човнів має реалізацію функцією `userprofboatset_change_view`, яка використовує клас форми `UserprofboatsetForm` та працює з добіркою човнів з номером `userprofboatsetId`.

Подання заявки на придбання човна має реалізацію функцією `boatsale_add_view`, де номер човна `boatId` є параметром.

Відтворення переліку заявок на придбання човна має реалізацію функцією `own_userprof_boatsales_view`, яка за користувачем, який авторизувався, вибирає його заявки на придбання човна і впорядковує їх за параметром `sortorder`: 0 – за датою подання заявки на придбання човна з нових, 1 – за датою подання заявки на придбання човна від старих.

Відтворення заявок на придбання човна має реалізацію функцією `boat_boatsales_view`, параметр якої `boatId` задає номер човна, а параметр `sortorder` може мати значення 0 для впорядкування за датою подання заявки на

придбання човна з нових, 1 для впорядкування за датою подання заявки на придбання човна від старих.

Вподобання човна з оголошення та відхилення вподобання човна з оголошення має реалізацію функцією `boat_like`, параметр якої `boatId` задає номер човна, а вподобання виконується тоді, коли воно відсутнє за цим користувачем для цього човна, а коли таке вже було зроблено, то вподобання навпаки скасовується.

Додавання човна до добірки та відхилення човна з добірки має реалізацію функцією `userprofboatset_add_boat_view`, де параметр `userprofboatsetId` задає номер добірки човнів, а параметр `boatId` задає номер човна, при цьому якщо добірка човнів не належить користувачу, який авторизувався, то обробка не виконується. Якщо човен є в добірці, то він вилучається.

Відтворення переліку добірок користувача має реалізацію функцією `own_userprofboatsets_view` за авторизованим обліковим записом.

Вилучення власної добірки човнів має реалізацію функцією `userprofboatset_delete_view`, яка має параметр `userprofboatsetId`, що задає номер добірки човнів. Вилучення реалізується тільки, якщо авторизований користувач створив цю добірку човнів.

Клас `Userprof` реалізує в структурі Django-застосунку модель профілю користувача, який може бути і клієнтом, і продавцем човнів.

Поле `user` класу є полем облікового запису користувача в Django, за яким працює цей користувач застосунку підтримки продажу човнів, тому є об'єктом класу `ForeignKey` на основі класу моделі `User`.

Поле `surandname` класу є полем повного імені та прізвища профілю користувача застосунку підтримки продажу човнів, тому є об'єктом класу `CharField`.

Поле `pic` класу є полем профільної фотографії користувача застосунку підтримки продажу човнів, тому є об'єктом класу `ImageField`.

Клас `Boattype` реалізує в структурі Django-застосунку модель типів човнів, які можуть їм присвоюватися в конкретних оголошеннях. Поле `boattypetitle` класу є полем назви типу човна, тому є об'єктом класу `CharField`.

Клас `Boatmanufacturer` реалізує в структурі Django-застосунку модель компаній-виробників човнів.

Поле `manufacturertitle` класу є полем назви компанії-виробника човна, тому є об'єктом класу `CharField`.

Поле `webaddress` класу є полем адреси вебсайту компанії-виробника човна, тому є об'єктом класу `CharField`.

Клас `Locationregion` реалізує в структурі Django-застосунку модель регіонів, за якими укрупнюється розташування міст, де можуть знаходитись човни. Поле `regiontitle` класу є полем назви регіону, за яким укрупнюється розташування міст, де можуть знаходитись човни, тому є об'єктом класу `CharField`.

Клас `Location` реалізує в структурі Django-застосунку модель міст, де можуть знаходитись човни під час продажу.

Поле `locationregion` класу є полем регіону, в якому це місто, де може знаходитись човен на продажу, знаходиться, тому є об'єктом класу `ForeignKey` на основі класу моделі `Locationregion`.

Поле `centertitle` класу є полем назви міста, де можуть знаходитись човни під час продажу, тому є об'єктом класу `CharField`.

Клас `Boatmodel` реалізує в структурі Django-застосунку модель моделей човнів.

Поле `boatmanufacturer` класу є полем компанії-виробника човнів, що виготовляє цю модель човна, тому є об'єктом класу `ForeignKey` на основі класу моделі `Boatmanufacturer`.

Поле `modeltitle` класу є полем самої назва моделі човна, тому є об'єктом класу `CharField`.

Поле `boattype` класу є полем типу човна, який представляє ця модель човна, тому є об'єктом класу `ForeignKey` на основі класу моделі `Boattype`.

Клас `Boat` реалізує в структурі Django-застосунку модель оголошень про продаж човнів.

Поле `dealerprof` класу є полем профілю власника човна, яким він представлений як користувач застосунку підтримки продажу човнів, тому є об'єктом класу `ForeignKey` на основі класу моделі `Userprof`.

Поле `adtitle` класу є полем назви оголошення про продаж човну, тому є об'єктом класу `CharField`.

Поле `location` класу є полем міста, де знаходиться човен, який в цьому оголошенні пропонується для придбання, тому є об'єктом класу `ForeignKey` на основі класу моделі `Location`.

Поле `boatmodel` класу є полем моделі човна, тому є об'єктом класу `ForeignKey` на основі класу моделі `Boatmodel`.

Поле `builtyear` класу є полем року виготовлення човна, який в оголошенні пропонується для придбання, тому є об'єктом класу `PositiveSmallIntegerField`.

Поле `regplate` класу є полем реєстраційного номера човна, який в оголошенні пропонується для придбання, тому є об'єктом класу `CharField`.

Поле `width` класу є полем ширини корпусу човна, який в оголошенні пропонується для придбання, тому є об'єктом класу `PositiveSmallIntegerField`.

Поле `length` класу є полем довжини корпусу човна, який в оголошенні пропонується для придбання, тому є об'єктом класу `PositiveSmallIntegerField`.

Поле `conditiontype` класу є полем стану човна, який в оголошенні пропонується для придбання, тому є об'єктом класу `CharField`, який здатен підтримувати роботу на основі відповідних кортежів, що встановлюють відповідність між числовим значенням стану та його описом.

Поле `boatcomment` класу є полем опису для оголошення човна, який в оголошенні пропонується для придбання, тому є об'єктом класу `TextField`.

Поле `price` класу є полем вартості придбання човна, який пропонується в оголошенні, тому є об'єктом класу `DecimalField`.

Поле `pricerescot` класу є полем рекомендованої вартості продажу човна за запропонованим методом прогнозування вартості продажу човна, тому є об'єктом класу `DecimalField`.

Поле `hulltype` класу є полем типу корпусу човна, який в оголошенні пропонується для придбання, тому є об'єктом класу `CharField`, який здатен підтримувати роботу на основі відповідних кортежів, що встановлюють відповідність між числовим значенням типу корпусу човна та його описом.

Поле `watercapacity` класу є полем водоемності човна, який в оголошенні пропонується для придбання, тому є об'єктом класу `PositiveIntegerField`.

Поле `hullmaterial` класу є полем матеріалу, з якого виготовлений корпус човна, який в оголошенні пропонується для придбання, тому є об'єктом класу `ForeignKey` на основі класу моделі `Buildmaterial`.

Поле `deckmaterial` класу є полем матеріалу, з якого виготовлена палуба човна, який в оголошенні пропонується для придбання, тому є об'єктом класу `ForeignKey` на основі класу моделі `Buildmaterial`.

Поле `showernumber` класу є полем кількості душових кабін на човні, який в оголошенні пропонується для придбання, тому є об'єктом класу `PositiveSmallIntegerField`.

Поле `toiletnumber` класу є полем кількості туалетів на човні, який в оголошенні пропонується для придбання, тому є об'єктом класу `PositiveSmallIntegerField`.

Поле `anchorwinch` класу є полем текстового опису особливостей якірної лебідки човна, який в оголошенні пропонується для придбання, тому є об'єктом класу `CharField`.

Поле `bilgerump` класу є полем опису особливостей трюмного насосу човна, який в оголошенні пропонується для придбання, тому є об'єктом класу `CharField`.

Поле `covers` класу є полем особливості покриття човна, який в оголошенні пропонується для придбання, тому є об'єктом класу `CharField`

Поле `entertainment` класу є полем системи для розваг (музичної системи) човна, який в оголошенні пропонується для придбання, тому є об'єктом класу `CharField`.

Поле `epirb` класу є полем служби радіозв'язку човна, який в оголошенні пропонується для придбання, що застосовується під час надзвичайних ситуацій, тому є об'єктом класу `CharField`.

Поле `lifejackets` класу є полем кількості рятувальних жилетів на човні, який в оголошенні пропонується для придбання, тому є об'єктом класу `PositiveSmallIntegerField`.

Поле `navigation` класу є полем навігаційної системи човна, який в оголошенні пропонується для придбання, тому є об'єктом класу `CharField`.

Поле `radiocommunication` класу є полем системи радіокомунікації човна, який в оголошенні пропонується для придбання, тому є об'єктом класу `CharField`.

Поле `enginefuelcapacity` класу є полем обсягу палива, який здатен вмістити човен, який в оголошенні пропонується для придбання, що використовується для роботи двигуна, тому є об'єктом класу `PositiveSmallIntegerField`.

Поле `enginehours` класу є полем кількості годин, яку за повного баку в середньому може працювати човен, який в оголошенні пропонується для придбання, тому є об'єктом класу `DecimalField`.

Поле `enginehorsepowers` класу є полем кількості кінських сил двигуна човна, який в оголошенні пропонується для придбання, тому є об'єктом класу `DecimalField`.

Поле `enginereleaseyear` класу є полем року виготовлення поточного встановленого двигуна на човні, який в оголошенні пропонується для придбання, тому є об'єктом класу `PositiveSmallIntegerField`.

Поле `displacement` класу є полем кількості виконаних заміन двигуна човна, який в оголошенні пропонується для придбання, тому є об'єктом класу `PositiveSmallIntegerField`.

Поле `enginemade` класу є полем основного матеріалу, який використовується при роботі двигуна човна, який в оголошенні пропонується для придбання, тому є об'єктом класу `CharField`, який здатен підтримувати роботу на основі відповідних кортежів, що встановлюють відповідність між числовим значенням матеріалу та його описом.

Поле `enginefueltype` класу є полем типу палива, необхідного для роботи двигуна човна, який в оголошенні пропонується для придбання, тому є об'єктом класу `CharField`, який здатен підтримувати роботу на основі відповідних кортежів, що встановлюють відповідність між числовим значенням типу палива та його описом.

Поле `state` класу є полем стану оголошення про продаж човна, тому є об'єктом класу `CharField`, який здатен підтримувати роботу на основі відповідних кортежів, що встановлюють відповідність між числовим значенням стану оголошення про продаж човна та його описом.

Поле `creationdate` класу є полем дати створення оголошення про продаж човна, тому є об'єктом класу `DateField`.

Поле `buildmaterials` класу є полем списку матеріалів, використаних при виробництві човна, тому є об'єктом класу `ManyToManyField` на основі класу моделі `Buildmaterial`.

Поле `userviews` класу є полем списку користувачів застосунку підтримки продажу човнів, які переглянули оголошення про продаж човну, тому є об'єктом класу `ManyToManyField` на основі класу моделі `Userprof`.

Поле `userlikes` класу є полем списку користувачів застосунку підтримки продажу човнів, які висловили вподобання човну, тому є об'єктом класу `ManyToManyField` на основі класу моделі `Userprof`.

Клас `Userprofboatset` реалізує в структурі Django-застосунку модель користувацьких добірок човнів.

Поле `userprof` класу є полем профілю користувача застосунку підтримки продажу човнів, який створив цю добірку, тому є об'єктом класу `ForeignKey` на основі класу моделі `Userprof`.

Поле `setname` класу є полем назви користувацької добірки човнів, тому є об'єктом класу `CharField`.

Поле `boats` класу є полем переліку човнів, який входять у цю користувацьку добірку, тому є об'єктом класу `ManyToManyField` на основі класу моделі `Boat`.

Клас `Boatpic` реалізує в структурі Django-застосунку модель фотографій човнів.

Поле `boat` класу є полем човну, який показаний на фотографії, тому є об'єктом класу `ForeignKey` на основі класу моделі `Boat`.

Поле `feature` класу є полем, яке позначає, це основна фотографія для представлення оголошення про продаж човна чи це додаткова фотографія, тому є об'єктом класу `BooleanField`.

Поле `pic` класу є полем фотографії човна, тому є об'єктом класу тому є об'єктом класу.

Клас `Buildmaterial` реалізує в структурі Django-застосунку модель матеріалів, з яких можуть виготовлятися човни.

Поле `biuldmaterialtitle` класу є полем назви матеріалу, з якого можуть виготовлятися човни, тому є об'єктом класу `CharField`.

Клас `Boatsale` реалізує в структурі Django-застосунку модель заявок на придбання човнів.

Поле `boat` класу є полем човна, заявку на придбання якого залишив користувач застосунку підтримки продажу човнів, тому є об'єктом класу `ForeignKey` на основі класу моделі `Boat`.

Поле `userprof` класу є полем користувача застосунку підтримки продажу човнів, який створив цю заявку на його придбання, тому є об'єктом класу `ForeignKey` на основі класу моделі `Userprof`.

Поле `applicationdate` класу є полем дати, часу подання заявки на придбання човна, створеної користувачем застосунку підтримки продажу човнів, тому є об'єктом класу `DateTimeField`.

Поле `comment` класу є полем коментаря до заявки на придбання човна, створеної користувачем застосунку підтримки продажу човнів, тому є об'єктом класу `CharField`.

Поле `state` класу є полем стану заявки на придбання човна, створеної користувачем застосунку підтримки продажу човнів, тому є об'єктом класу `CharField`, який здатен підтримувати роботу на основі відповідних кортежів, що встановлюють відповідність між числовим значенням стану заявки на придбання човна та його описом.

Клас `Dealerquestion` реалізує в структурі Django-застосунку модель запитань і відповідей між покупцями і продавцями човнів.

Поле `dealerprof` класу є полем продавця човна, якому направлено запитання або який створив відповідь на запитання покупця човна, тому є об'єктом класу `ForeignKey` на основі класу моделі `Userprof`.

Поле `userprof` класу є полем покупця човна, якому направлено відповідь на запитання або який створив запитання до покупця човна, тому є об'єктом класу `ForeignKey` на основі класу моделі `Userprof`.

Поле `dealerque` класу є полем, що визначає, це запитання покупця човна до його продавця чи це відповідь продавця човна на запитання покупця, тому є об'єктом класу `BooleanField`.

Поле `questiontxt` класу є полем тексту запитання або відповіді між покупцем і продавцем човнів, тому є об'єктом класу `CharField`.

Поле `questiondate` класу є полем дати, часу створення запитання або відповіді між покупцем і продавцем човнів, тому є об'єктом класу `DateTimeField`.

4.3 Висновки за розділом 4

Реалізацію програми підтримки продажу човнів у цьому розділі завершено з приведенням опису тих структурних одиниць вебзастосунку, які було реалізовано в результаті.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Опис програми

Програма призначена для підтримки виставлення власних нових або вживаних човнів для продажу шляхом створення відповідних оголошень, створення, підтвердження або відхилення заявок на придбання човнів, пошуку необхідних човнів.

5.2 Експлуатація програми

Користувач програми підтримки продажу човнів може використовувати в першу чергу будь-які засоби, доступні йому з основного меню.

З першого розділу меню доступний перелік оголошень про продаж човнів, які створив користувач та за якими він продає човни. Якщо користувач ще не продає човнів, то йому відображається сторінка з пропозицією опублікувати оголошення. Отримати доступ до публікації оголошення про продаж човна можна і з меню з розділу «Продати».

З другого розділу меню можна переглянути ті оголошення про продаж човна, які були вподобані користувачем. Фактично ж вподобання видні на кожній сторінці, де відбувається робота з човнами. На кожній такій сторінці є інформація з кількістю вподобань та переглядів, а відповідно можна вподобати або відхилити вподобання за оголошенням про продаж човна.

З третього розділу меню доступна можливість переглянути всі добірки човнів, які створив авторизований у програмі користувач. Тоді можна натиснути на потрібну та відкрити сторінку такої добірки (рис. 5.1). З цієї сторінки можна відредагувати добірку човнів, можна вилучити цю добірку човнів, переглянути перелік човнів у ній, а за кожним човном отримати

інформацію про те, яка кількість переглядів, уподобань є у човна, вилучити човен з добірки, якщо це необхідно.

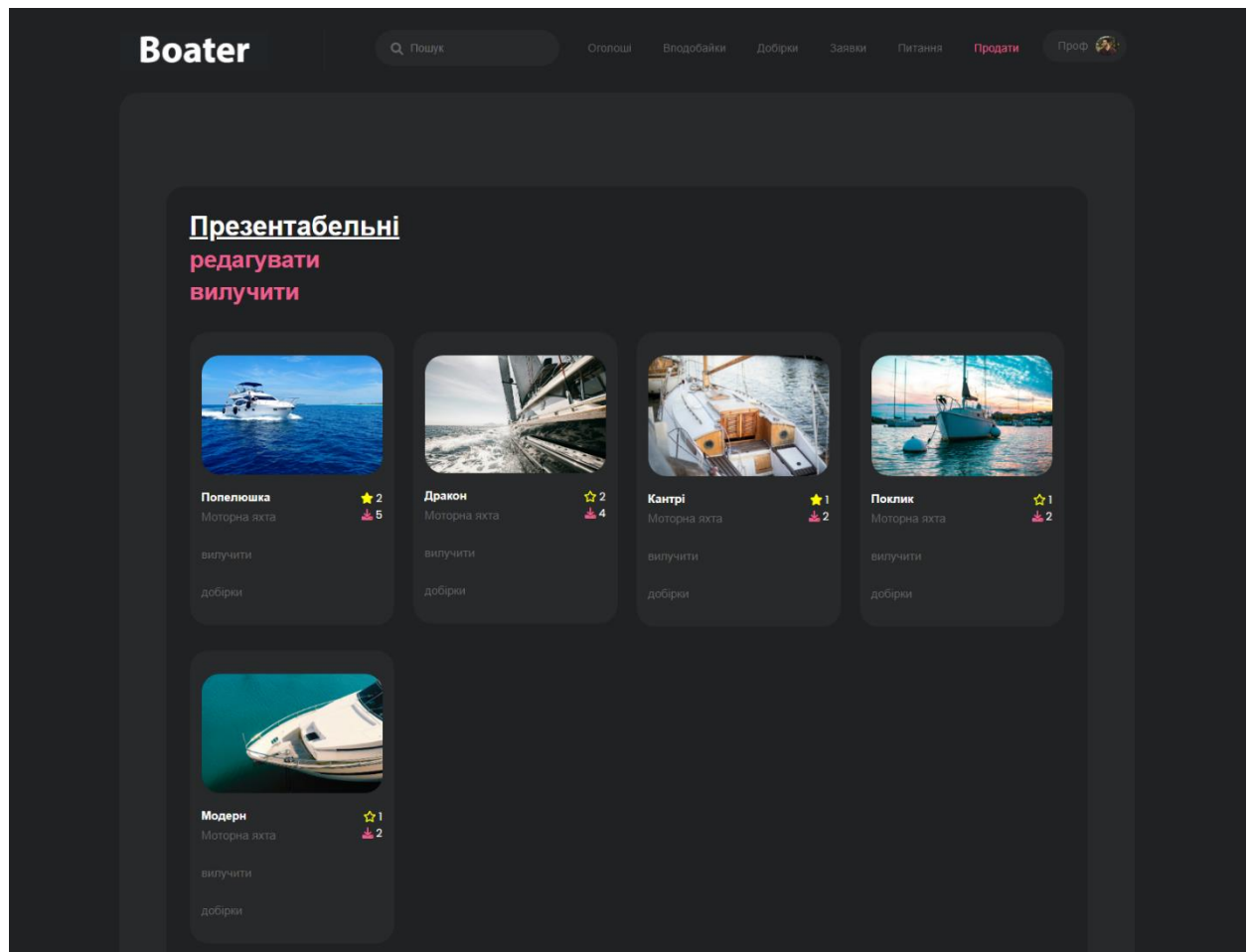


Рисунок 5.1 – Сторінка добірки човнів

Для того щоб додати човен до добірки човнів, необхідно натиснути на посилання з назвою списки, а тоді відобразиться випадаючий перелік, звідки можна обрати потрібну добірку, переглянути таким чином, в якій саме добірці присутній човен. Можна також за допомогою відповідного посилання з цієї добірки човнів відповідний човен вилучити. Також можна переглянути про конкретний човен, в який добірках авторизованого користувача він присутній, додати до іншої добірки човен, можна зі сторінки оголошення про продаж човна. На цій сторінці розташована вся інформація про човен. Можна також подати заявку на придбання човна (рис. 5.2).

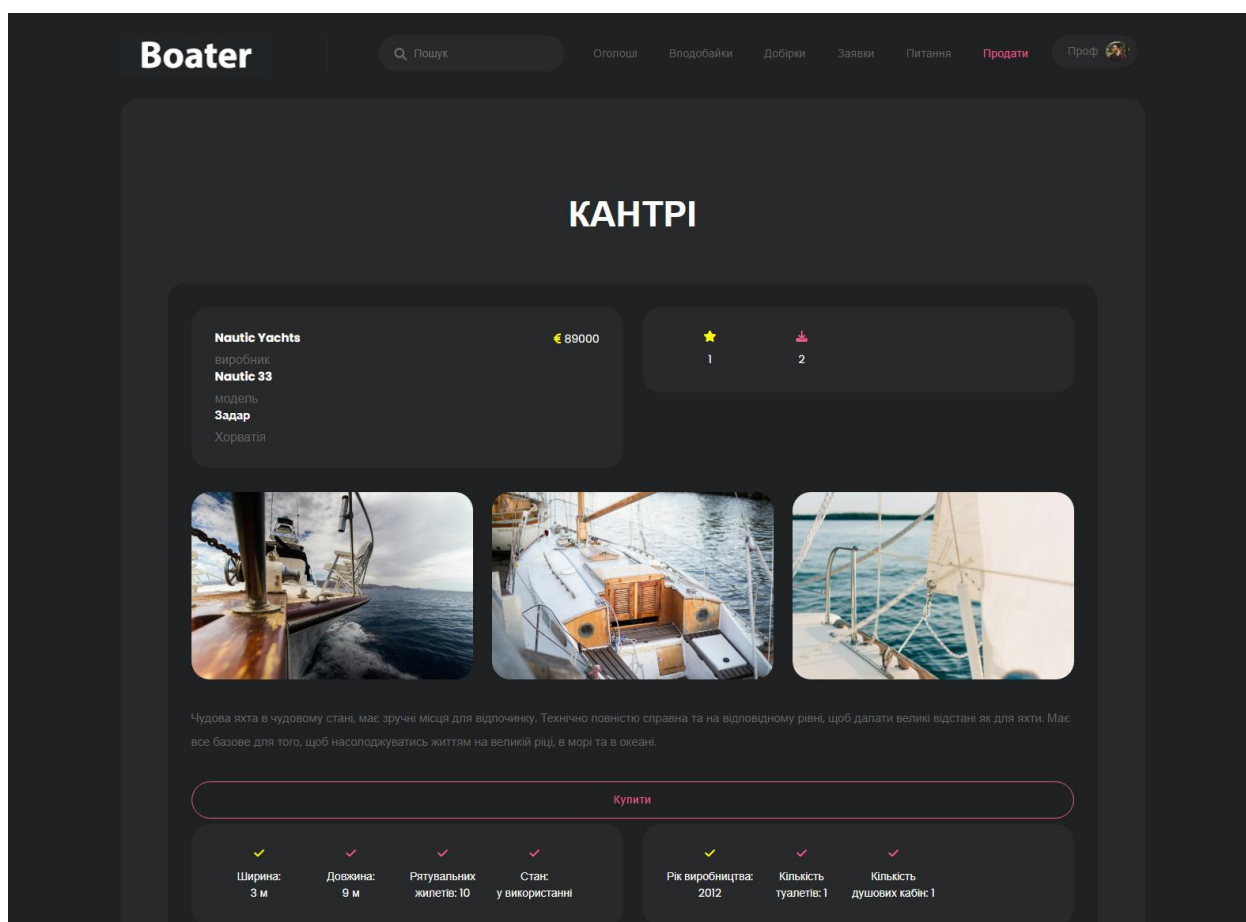


Рисунок 5.2 – Сторінка оголошення про продаж човна

Зі сторінки оголошення про продаж човна можна також отримати інформацію про власника човна, написати йому запитання, перейшовши таким чином до розмови з ним.

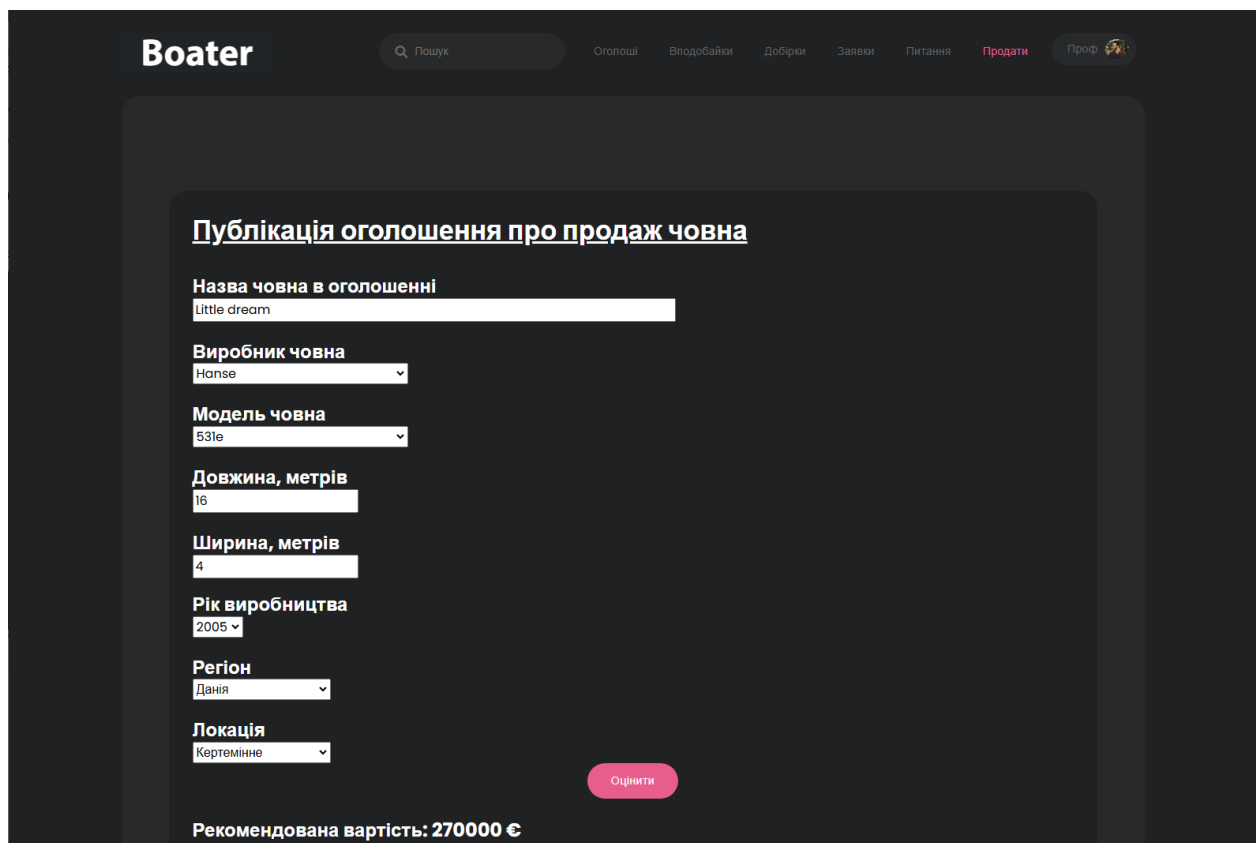
Відповідно подивитись на ті заявки на придбання човнів, які залишив користувач, можна з меню в розділі «Заявки». Для роботи з іншої сторони за заявками, які подані власнику, сам власник може обрати розділ оголошень, звернувшись до власних оголошень про продаж човнів, а звідти за кожним оголошенням доступна можливість перейти до переліку поданих користувачами заявок на придбання відповідного човна.

Розділ меню «Питання» відкриває сторінку, яка складається з двох варіантів, перемикається між якими можна вибором відповідного варіанта: обираючи перелік власників, яким написав користувач, або обираючи перелік

тих користувачів, які написали авторизованому користувачу як власнику (за умов наявності оголошень про продаж човна у користувача).

З профілю користувача можна не тільки переглянути дані про нього, але також і вийти з авторизації, перейти до редагування даних.

Під час публікації нового оголошення про продаж власного човна (рис. 5.3) потрібно задати всі необхідні дані, включаючи зокрема і матеріали, з яких виготовлений човен, додати необхідні фотографії, але верхня частина включає визначення мінімальної частини оголошення, включаючи довжину човна, його ширину, рік виробництва, регіон та локацію його розташування, що дозволяє натиснути на кнопку для оцінювання вартості продажу такого човна. Після цього натискання інформація на сторінці доповнюється рекомендованою ціною.



The screenshot shows the 'Boater' website interface. At the top, there is a navigation bar with a search bar and links for 'Оголоші', 'Вподобайки', 'Добірки', 'Заявки', 'Питання', 'Продати', and 'Проф'. The main content area is titled 'Публікація оголошення про продаж човна'. Below this title, there is a form with several fields: 'Назва човна в оголошенні' (Little dream), 'Виробник човна' (Hanse), 'Модель човна' (531e), 'Довжина, метрів' (16), 'Ширина, метрів' (4), 'Рік виробництва' (2005), 'Регіон' (Данія), and 'Локація' (Кертемінне). A pink button labeled 'Оцінити' is positioned below the form fields. At the bottom of the form, it says 'Рекомендована вартість: 270000 €'.

Рисунок 5.3 – Публікація оголошення про продаж човна

5.3 Експериментальне дослідження

Початково вибірка [9], яка використана для експериментального дослідження прогнозування вартості продажу човнів, складається з наступних ознак:

- тип човна;
- виробник човна;
- додаткові характеристики типу човна;
- рік виготовлення човна;
- довжина човна;
- ширина човна;
- основний матеріал, з якого виготовлений човен;
- локація, де човен розташований;
- кількість переглядів оголошення про продаж відповідного човна за останні 7 днів.

Під час експериментального дослідження прогнозування вартості продажу човнів розглянуто запропонований метод та його альтернативи у вигляді окремого дерева рішень та методу опорних векторів (табл. 5.1).

Для оцінювання та порівняння отриманих результатів використано як показники:

- показник Mean Absolute Error (MAE);
- показник Mean Squared Error (MSE);
- показник Root Mean Squared Error (RMSE).

Дерева рішень найкращий результат прогнозування вартості продажу човнів дозволили отримати у тому випадку, коли на вхід подано 7 вхідних ознак, наявних у вибірці та перетворених таким чином, як це було зроблено у запропонованому методі. Найкращі результати в такому випадку отримано і за показником RMSE, і MSE. Проте за показником MAE найкращий результат було отримано тоді, коли дерево рішень має 5 вхідних ознак. Окрім того

дерева рішень мали кращі результати прогнозування вартості продажу човнів порівняно з методом опорних векторів.

Таблиця 5.1 – Прогнозування вартості продажу човнів

Модель	MAE	RMSE	MSE
Дерево рішень (9 ознак)	0,001533	0,010117	0,000102
Дерево рішень (8 ознак)	0,001511	0,010189	0,000104
Дерево рішень (7 ознак)	0,001346	0,00774	5,99122E-05
Дерево рішень (6 ознак)	0,00131	0,007839	6,576724E-05
Дерево рішень (5 ознак)	0,001298	0,008052	6,483627E-05
Дерево рішень (4 ознаки)	0,001305	0,008357	6,984261E-05
Дерево рішень (3 ознаки)	0,001347	0,008527	7,270321E-05
Дерево рішень (2 ознаки)	0,001556	0,008009	6,41441E-05
Метод опорних векторів	0,092738	0,093114	0,00867

При використанні запропонованого методу застосування Random Forest призвело до зменшення показника MAE в першу чергу (табл. 5.2). Ансамбль дерев рішень побудований з 500 дерев. Найкращий результат отримано за показником MAE при 5 вхідних ознаках зі значенням 0.001075. Порівняно з найкращим результатом дерев рішень це зменшило показник MAE на 17 %.

Набір з 5 вхідних ознак при цьому складається з наступних:

- довжина човна;
- ширина човна;
- рік виготовлення човна;
- тип човна;
- локація, де човен розташований.

Таблиця 5.2 – Прогнозування вартості продажу човна на основі запропонованого методу в різних варіаціях вхідних ознак

Модель	MAE	RMSE	MSE
Random Forest за запропонованим методом (9 ознак)	0,001141	0,008006	6,409851E-05
Random Forest за запропонованим методом (8 ознак)	0,001141	0,007953	6,324723E-05
Random Forest за запропонованим методом (7 ознак)	0,001127	0,007824	6,121165E-05
Random Forest за запропонованим методом (6 ознак)	0,001119	0,007688	5,911011E-05
Random Forest за запропонованим методом (5 ознак)	0,001075	0,007585	5,752831E-05
Random Forest за запропонованим методом (4 ознаки)	0,001113	0,007574	5,737047E-05
Random Forest за запропонованим методом (3 ознаки)	0,001131	0,007796	6,07748E-05
Random Forest за запропонованим методом (2 ознаки)	0,001491	0,009493	9,01143E-05

5.4 Висновки за розділом 5

Реалізацію програмного забезпечення підтримки продажу човнів підтверджено на основі перевірки функціональних можливостей та відповідного виконаного опису логіки роботи. Проведено експериментальне дослідження прогнозування вартості продажу човнів, що підтвердило підвищення точності зокрема на основі показника MAE за запропонованим методом. Порівняння альтернатив виконано з методом опорних векторів та деревом рішень.

ВИСНОВКИ

За результатами виконання роботи розроблено програмне забезпечення підтримки продажу човнів для забезпечення рішень стосовно визначення вартості продажу на основі даних.

Для розробки програми підтримки продажу човнів було використано:

- мову програмування Python та Python-пакети для розробки, що забезпечили реалізацію запропонованого методу прогнозування вартості продажу човна, роботу з даними;
- фреймворк веброзробки Django;
- систему керування базами даних MySQL.

Створена програма призначена для підтримки виставлення власних нових або вживаних човнів для продажу шляхом створення відповідних оголошень, створення, підтвердження або відхилення заявок на придбання човнів, пошуку необхідних користувача човнів. Користувачам надається можливість створювати власні добірки човнів, щоб повертатись до раніше відібраних. Проєктування програми виконано за результатами огляду вебзастосунків підтримки продажу човнів, в якому в основному було розглянуто програми Boatsales.com.au, AutoRia, Boat24, враховано їх основні можливості та спосіб підтримки цих можливостей, визначено основні недоліки, а також як результат встановлено ті характеристики програми, які мають бути відмінностями програми, що розробляється. Виключними особливостями виконаної розробки програми підтримки продажу човнів є наявність можливості створення добірок човнів, рекомендавання вартості продажу човна для власника.

Наукова новизна роботи полягає в методі прогнозування вартості продажу човна, який на основі ознак оголошення про продаж човна визначає створення і навчання моделі як ансамблю дерев рішень Random Forest, що дозволяє провести рекомендаційне оцінювання вартості човна для формування підсумкового оголошення.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. New & Used Boat Sales – Find Boats For Sale Online – boatsales.com.au [Electronic resource]. – Access mode : <https://www.boatsales.com.au/>.
2. AUTO.RIA – Автобазар №1, купити та продати перевірене авто легко! [Електрон. ресурс]. – Режим доступу : <https://auto.ria.com/uk/>.
3. Boats for Sale – International Marketplace for Yachts & Boats [Electronic resource]. – Access mode : <https://www.boat24.com/en/>.
4. Spotfire. Demystifying the Random Forest Algorithm for Accurate Predictions [Electronic resource]. – Access mode : <https://www.spotfire.com/glossary/what-is-a-random-forest>.
5. How random forest works in data science? [Electronic resource]. – Access mode : <https://www.edcparis.edu/en/school-news/how-random-forest-works-data-science>.
6. Django at a glance. Django documentation. Django [Electronic resource]. – Access mode : <https://docs.djangoproject.com/en/5.1/intro/overview/>.
7. Installation – Laravel 11.x – The PHP Framework For Web Artisans [Electronic resource]. – Access mode : <https://laravel.com/docs/11.x>.
8. Free Template 579 Cyborg Gaming [Electronic resource]. – Access mode : <https://templatemo.com/tm-579-cyborg-gaming>.
9. Ali, M. R. Boats Listing Dataset [Electronic resource] / M. R. Ali. – Access mode : <https://www.kaggle.com/datasets/muhammadrayanali/traintest>.

ДОДАТОК А
Технічне завдання

Вступ

Човни не мають такого розповсюдження як автомобілі, проте це окрема сфера, яка заслуговує уваги, тим більше зважаючи на кількість водойм, які є в Україні. Такі човни можуть мати дуже різні типи, розміри. Відповідно дуже різних користувачів з дуже різними потребами потрібно задовольнити відповідними програмними засобами, враховуючи особливості човнів порівняно з автомобілями, надаючи гнучкий пошук для цього. Проте важливим кроком під час виставлення власного човна для продажу є визначення умов такого продажу, з яких головною є встановлення вартості човна. Зважаючи на специфіку ринку та потребу в нових продавцях човнів, тобто розширенні ринку, необхідно в обов'язковому порядку такий програмний засіб забезпечити рекомендаційною функцією, яка б могла запропонувати продавцю адекватну вартість для його човна.

A.1 Підстави для розробки

На підставі наказу № 398 від 7 жовтня 2024 р., який був виданий у Національному університеті «Запорізька політехніка», де і ведеться розробка в межах даної дипломної кваліфікаційної роботи магістра, визначена тема роботи «Дослідження та програмна реалізація методів підтримки продажу човнів», що було деталізовано також виданим завданням на роботу.

A.2 Призначення розробки

Розробка має бути призначена для підтримки виставлення власних нових або вживаних човнів для продажу шляхом створення відповідних оголошень, створення, підтвердження або відхилення заявок на придбання човнів, пошуку необхідних човнів.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Для реалізації комплексної роботи програмного забезпечення підтримки продажу човнів необхідно забезпечити такі функціональні можливості:

- реєстрація та авторизація користувачів застосунку підтримки продажу човнів;
- пошук човнів для придбання;
- створення оголошення про продаж човна з визначенням рекомендованої ціни продажу човна;
- відтворення переліку оголошень про продаж човнів у користувача, у тому числі власні;
- перегляд оголошення про продаж човна;
- редагування оголошення про продаж човна;
- вподобання човна з оголошення;
- відхилення вподобання човна з оголошення;
- відтворення переліку човнів, уподобаних користувачем;
- створення добірки човнів;
- редагування описових даних добірки човнів;
- додавання човна до добірки;
- відхилення човна з добірки;
- подання заявки на придбання човна;
- підтвердження заявки на придбання човна;
- відхилення заявки на придбання човна;
- відтворення переліку заявок на придбання човна;
- відтворення власних заявок на придбання човна;
- відтворення переліку човнів з добірки;
- відтворення переліку добірок користувача;
- адресування запитання продавцю човна;

- відправлення відповіді на запитання продавцю човна;
- відтворення переліку всіх покупців човнів, які поставили запитання;
- відтворення переліку всіх продавців, яким поставив запитання покупець;
- відтворення запитань між продавцем та покупцем човна;
- перегляд профілю користувача програми підтримки продажу човнів;
- редагування власних даних профілю;
- вилучення власної добірки човнів.

A.3.2 Умови експлуатації

Експлуатація програмного забезпечення підтримки продажу човнів має забезпечуватися виконанням ряду вимог до серверної частини:

- забезпечення вебсерверу з підтримкою інтерпретатора мови програмування Python 3;
- налаштування Django-застосунку для виконання на сервері з реалізацією міграції для забезпечення необхідної структури бази даних за допомогою підтримки системи керування базою даних MySQL;
- інсталювані Python-пакети Numpy, Pandas, Scikit-learn.

A.3.3 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення підтримки продажу човнів має забезпечуватися серверною частиною з такими мінімальними параметрами технічних засобів:

- двоядерний процесор;
- 4 Гб оперативної пам'яті;
- 10 Гб простору на жорсткому диску, що необхідно для збереження даних самого вебсайту та фотографій човнів, файлів вебзастосунку, бази

даних: в залежності від якості, кількості фотографій можливе подальше збільшення мінімальних вимог.

А.4 Порядок контролю та приймання

Робота має виконуватися у відповідності з затвердженим завданням і контролюється керівником. Отримані результати роботи представляються в пояснювальній записці.

Завершальним етапом виконання та відповідно контролю і приймання дипломної кваліфікаційної роботи магістра є її захист, який виконується перед екзаменаційною комісією.

ДОДАТОК Б
Текст програми

```

from .models import Userprof
from .models import Boatttype
from .models import Boatmanufacturer
from .models import Locationregion
from .models import Location
from .models import Boatmodel
from .models import Boat
from .models import Userprofboatset
from .models import Boatpic
from .models import Buildmaterial
from .models import Boatsale
from .models import Dealerquestion
from django.core.paginator import Paginator
from django.shortcuts import render

def own_userprof_boats_view(request, sortorder = 0):
    if not request.user.is_authenticated:
        render(request, "boater/oboats.html",
               {"boats": None,
                "boatsets": []
               })
    try:
        userprofs =
Userprof.objects.select_related("user").all()
        userprof = userprofs.get(user = request.user)
    except Userprof.DoesNotExist:
        userprof = None
        return render(request, "boater/oboats.html",
                      {"boats": None,
                       "boatsets": []
                      })
    boats = Boat.objects.select_related("dealerprof",
                                         "boatmodel",
                                         "boatmodel__boatmanufacturer",
                                         "boatmodel__boatttype",
                                         "location").prefetch_related("userviews",
                                                                        "userlikes").all()
    boats = boats.filter(dealerprof = userprof)
    from django.db.models import Count
    boats = boats.annotate(userviews_count =
Count("userviews"))
    boats = boats.annotate(userlikes_count =
Count("userlikes"))
    if sortorder == 0:
        boats = boats.order_by("-creationdate")
    elif sortorder == 1:
        boats = boats.order_by("creationdate")
    elif sortorder == 2:
        boats = boats.order_by("userviews_count")
    elif sortorder == 3:
        boats = boats.order_by("-userviews_count")
    elif sortorder == 4:

```

```

        boats = boats.order_by("userlikes_count")
    elif sortorder == 5:
        boats = boats.order_by("-userlikes_count")
    elif sortorder == 6:
        boats = boats.order_by("adtitle")
    elif sortorder == 7:
        boats = boats.order_by("-adtitle")
    boatsets = []
    userprofile = request.user
    userprofile = Userprof.objects.get(user = userprofile)
    for boat in boats:
        if userprofile in boat.userlikes:
            boat.wasliked = True
        else:
            boat.wasliked = False
        if userprofile in boat.userviews:
            boat.wasviewed = True
        else:
            boat.wasviewed = False
    boatsets =
    Userprofboatset.objects.prefetch_related("boats").all()
    boatsets = boatsets.filter(userprof = userprofile)
    for boat in boats:
        boat.boatsets = []
        for boatset in boatsets:
            if boat in boatset.boats:
                boat.boatsets.append(boatset)
            else:
                boatset.boats = []
    return render(request, "boater/oboats.html",
                  {"boats": boats,
                   "boatsets": boatsets
                  })

def boat_detail_view(request, boatId):
    try:
        boats = Boat.objects.select_related("boatmodel",
                                             "boatmodel__boattype",
                                             "boatmodel__boatmanufacturer",
                                             "location",
                                             "location__locationregion",
                                             "dealerprof",
                                             "dealerprof__user",
                                             "hullmaterial",
                                             "deckmaterial").prefetch_related("buildmaterials",
                                             "userviews",
                                             "userlikes").all()
        boat = boats.get(id = boatId)
    except Boat.DoesNotExist:
        boat = None

```

```

        return render(request, "boater/boat.html", {"boat":
None, "boatsets": None })
        boatsets = []
        if boat:
            boat.userviews_count = boat.userviews.count()
            boat.userlikes_count = boat.userlikes.count()
            boat.pics = Boatpic.objects.filter(boat = boat)
            if request.user.is_authenticated:
                userprofile = request.user
                userprofile = Userprof.objects.get(user =
userprofile)
                boat_view = Boatview(boat = boat, userprof =
userprofile)
                boat_view.save()
                if userprofile in boat.userlikes:
                    boat.wasliked = True
                else:
                    boat.wasliked = False
                if userprofile in boat.userviews:
                    boat.wasviewed = True
                else:
                    boat.wasviewed = False
            boatsets =
Userprofboatset.objects.prefetch_related("boats").all()
            boatsets = boatsets.filter(userprof =
userprofile)
            for boatset in boatsets:
                if boat in boatset.boats:
                    boatset.boats = [boat.id]
                else:
                    boatset.boats = []
        return render(request, "boater/boat.html",
{"boat": boat,
"boatsets": boatsets
})

def userprof_boats_view(request, userprofName, sortorder =
0):
    try:
        userprofs =
Userprof.objects.select_related("user").all()
        userprof = userprofs.get(user__username =
userprofName)
    except Userprof.DoesNotExist:
        userprof = None
        return render(request, "boater/usboats.html",
{"boats": None,
"boatsets": []
})
    boats = Boat.objects.select_related("dealerprof",
"boatmodel",
"boatmodel__boatmanufacturer",

```

```

        "boatmodel__boattype",
        "location").prefetch_related("userviews",
        "userlikes").all()
boats = boats.filter(dealerprof = userprof)
from django.db.models import Count
boats = boats.annotate(userviews_count =
Count("userviews"))
boats = boats.annotate(userlikes_count =
Count("userlikes"))
if sortorder == 0:
    boats = boats.order_by("-creationdate")
elif sortorder == 1:
    boats = boats.order_by("creationdate")
elif sortorder == 2:
    boats = boats.order_by("userviews_count")
elif sortorder == 3:
    boats = boats.order_by("-userviews_count")
elif sortorder == 4:
    boats = boats.order_by("userlikes_count")
elif sortorder == 5:
    boats = boats.order_by("-userlikes_count")
elif sortorder == 6:
    boats = boats.order_by("adtitle")
elif sortorder == 7:
    boats = boats.order_by("-adtitle")
boatsets = []
if request.user.is_authenticated:
    userprofile = request.user
    userprofile = Userprof.objects.get(user =
userprofile)
    for boat in boats:
        if userprofile in boat.userlikes:
            boat.wasliked = True
        else:
            boat.wasliked = False
        if userprofile in boat.userviews:
            boat.wasviewed = True
        else:
            boat.wasviewed = False
    boatsets =
Userprofboatset.objects.prefetch_related("boats").all()
boatsets = boatsets.filter(userprof = userprofile)
for boat in boats:
    boat.boatsets = []
    for boatset in boatsets:
        if boat in boatset.boats:
            boat.boatsets.append(boatset)
        else:
            boatset.boats = []
return render(request, "boater/usboats.html",
{"boats": boats,
"boatsets": boatsets
})

```

```

def liked_boats_view(request, sortorder = 0):
    if not request.user.is_authenticated:
        render(request, "boater/lboats.html",
                {"boats": None,
                 "boatsets": []
                })
    try:
        userprofs =
Userprof.objects.select_related("user").all()
        userprof = userprofs.get(user = request.user)
    except Userprof.DoesNotExist:
        userprof = None
        return render(request, "boater/lboats.html",
                        {"boats": None,
                         "boatsets": []
                        })
    boats = Boat.objects.select_related("dealerprof",
                                         "boatmodel",
                                         "boatmodel__boatmanufacturer",
                                         "boatmodel__boattype",
                                         "location").prefetch_related("userviews",
"userlikes").all()
    boats = boats.all()
    from django.db.models import Count
    boats = boats.annotate(userviews_count =
Count("userviews"))
    boats = boats.annotate(userlikes_count =
Count("userlikes"))
    if sortorder == 0:
        boats = boats.order_by("-creationdate")
    elif sortorder == 1:
        boats = boats.order_by("creationdate")
    elif sortorder == 2:
        boats = boats.order_by("userviews_count")
    elif sortorder == 3:
        boats = boats.order_by("-userviews_count")
    elif sortorder == 4:
        boats = boats.order_by("userlikes_count")
    elif sortorder == 5:
        boats = boats.order_by("-userlikes_count")
    elif sortorder == 6:
        boats = boats.order_by("adtitle")
    elif sortorder == 7:
        boats = boats.order_by("-adtitle")
    boatsets = []
    userprofile = request.user
    userprofile = Userprof.objects.get(user = userprofile)
    liked_boats = []
    for boat in boats:
        if userprofile in boat.userlikes:
            if userprofile in boat.userviews:
                boat.wasviewed = True
            else:

```

```

        boat.wasviewed = False
        liked_boats.append(boat)
    boatsets =
Userprofboatset.objects.prefetch_related("boats").all()
    boatsets = boatsets.filter(userprof = userprofile)
    for boat in boats:
        boat.boatsets = []
        for boatset in boatsets:
            if boat in boatset.boats:
                boat.boatsets.append(boatset)
            else:
                boatset.boats = []
    return render(request, "boater/lboats.html",
                  {"boats": liked_boats,
                   "boatsets": boatsets
                  })

def history_boats_view(request, sortorder = 0):
    if not request.user.is_authenticated:
        render(request, "boater/hboats.html",
              {"boats": None,
               "boatsets": []
              })
    try:
        userprofs =
Userprof.objects.select_related("user").all()
        userprof = userprofs.get(user = request.user)
    except Userprof.DoesNotExist:
        userprof = None
        return render(request, "boater/hboats.html",
                      {"boats": None,
                       "boatsets": []
                      })
    boats = Boat.objects.select_related("dealerprof",
                                         "boatmodel",
                                         "location",
                                         "boatmodel__boatmanufacturer",

"boatmodel__boattype").prefetch_related("userviews",
                                         "userlikes").all()
    boats = boats.all()
    from django.db.models import Count
    boats = boats.annotate(userviews_count =
Count("userviews"))
    boats = boats.annotate(userlikes_count =
Count("userlikes"))
    if sortorder == 0:
        boats = boats.order_by("-creationdate")
    elif sortorder == 1:
        boats = boats.order_by("creationdate")
    elif sortorder == 2:
        boats = boats.order_by("userviews_count")

```



```

boats = Boat.objects.select_related("dealerprof",
    "boatmodel",
    "boatmodel__boatmanufacturer",
    "boatmodel__boatttype",
    "location").prefetch_related("userviews",
    "userlikes").all()
try:
    userprofboatsets =
Userprofboatset.objects.prefetch_related("boats").select_related
("userprof").all()
    userprofboatset = userprofboatsets.get(id =
boatsetId)
    userprofboatset.ids = []
    for boat in userprofboatset.boats:
        userprofboatset.ids.append(boat.id)
except Userprofboatset.DoesNotExist:
    userprofboatset = None
    return render(request, "boater/boatset.html",
        {"boats": None,
        "boatsets": []
        })
boats = boats.filter(id__in = userprofboatset.ids)
from django.db.models import Count
boats = boats.annotate(userviews_count =
Count("userviews"))
boats = boats.annotate(userlikes_count =
Count("userlikes"))
if sortorder == 0:
    boats = boats.order_by("-creationdate")
elif sortorder == 1:
    boats = boats.order_by("creationdate")
elif sortorder == 2:
    boats = boats.order_by("userviews_count")
elif sortorder == 3:
    boats = boats.order_by("-userviews_count")
elif sortorder == 4:
    boats = boats.order_by("userlikes_count")
elif sortorder == 5:
    boats = boats.order_by("-userlikes_count")
elif sortorder == 6:
    boats = boats.order_by("adtitle")
elif sortorder == 7:
    boats = boats.order_by("-adtitle")
boatsets = []
userprofile = request.user
userprofile = Userprof.objects.get(user = userprofile)
for boat in boats:
    if userprofile in boat.userlikes:
        boat.wasliked = True
    else:
        boat.wasliked = False
    if userprofile in boat.userviews:
        boat.wasviewed = True

```

```

        else:
            boat.wasviewed = False
        boatsets =
Userprofboatset.objects.prefetch_related("boats").all()
        boatsets = boatsets.filter(userprof = userprofile)
        for boat in boats:
            boat.boatsets = []
            for boatset in boatsets:
                if boat in boatset.boats:
                    boat.boatsets.append(boatset)
            else:
                boatset.boats = []
        return render(request, "boater/boatset.html",
            {"boats": boats,
             "boatsets": boatsets
            })

def search_boats_view(request):
    if request.user.is_authenticated:
        try:
            userprofs =
Userprof.objects.select_related("user").all()
            userprof = userprofs.get(user = request.user)
        except Userprof.DoesNotExist:
            userprof = None
        boat_models =
Boatmodel.objects.select_related("boatmanufacturer").all()
        boat_manufacturers = Boatmanufacturer.objects.all()
        boat_locations =
Location.objects.select_related("locationregion").all()
        boat_types = Boattype.objects.all()
        boat_materials = Buildmaterial.objects.all()
        boats = Boat.objects.select_related("dealerprof",
"boatmodel", "boatmodel__boatmanufacturer",
"boatmodel__boattype", "location").prefetch_related("userviews",
"userlikes").all()
        boat_data = {}
        boat_data["manufacturers"] = boat_manufacturers
        boat_manuf = request.GET.get('manuf')
        if boat_manuf:
            try:
                boat_manuf =
Boatmanufacturer.objects.get(manufacturertitle = boat_manuf)
            except Boatmanufacturer.DoesNotExist:
                boat_manuf = None
            render(request, "boater/searchboats.html",
                {"boats": [],
                 "boatsets": [],
                 "boatmodels": boat_models,
                 "boattypes": boat_types,
                 "boatlocations": boat_locations,
                 "boatmaterials": boat_materials,

```

```

        "manufacturers": boat_manufacturers
    })
    boats = boats.filter(boatmodel__boatmanufacturer =
boat_manuf)
    boat_data["boatmodels"] = boat_models
    boat_model = request.GET.get('model')
    if boat_model:
        try:
            boat_model = Boatmodel.objects.get(modeltitle =
boat_model)
        except Boatmodel.DoesNotExist:
            boat_model = None
            render(request, "boater/searchboats.html",
                {"boats": [],
                 "boatsets": [],
                 "boatmodels": boat_models,
                 "boattypes": boat_types,
                 "boatlocations": boat_locations,
                 "boatmaterials": boat_materials,
                 "manufacturers": boat_manufacturers
                })
            boats = boats.filter(boatmodel = boat_model)
            boat_data["boatlocations"] = boat_locations
            boat_location = request.GET.get('location')
            if boat_location:
                try:
                    boat_location =
Location.objects.get(centertitle = boat_location)
                except Location.DoesNotExist:
                    boat_location = None
                    render(request, "boater/searchboats.html",
                        {"boats": [],
                         "boatsets": [],
                         "boatmodels": boat_models,
                         "boattypes": boat_types,
                         "boatlocations": boat_locations,
                         "boatmaterials": boat_materials,
                         "manufacturers": boat_manufacturers
                        })
                    boats = boats.filter(location = boat_location)
            boat_data["boatmaterials"] = boat_materials
            boat_hullmaterial = request.GET.get('hullmaterial')
            if boat_hullmaterial:
                try:
                    boat_hullmaterial = int(boat_hullmaterial)
                    boat_hullmaterial =
Buildmaterial.objects.get(id = boat_hullmaterial)
                except Buildmaterial.DoesNotExist:
                    boat_hullmaterial = None
                    render(request, "boater/searchboats.html",
                        {"boats": [],
                         "boatsets": [],
                         "boatmodels": boat_models,

```

```

        "boattypes": boat_types,
        "boatlocations": boat_locations,
        "boatmaterials": boat_materials,
        "manufacturers": boat_manufacturers
    })
    boats = boats.filter(hullmaterial =
boat_hullmaterial)
    boat_deckmaterial = request.GET.get('deckmaterial')
    if boat_deckmaterial:
        try:
            boat_deckmaterial = int(boat_deckmaterial)
            boat_deckmaterial =
Buildmaterial.objects.get(id = boat_deckmaterial)
        except Buildmaterial.DoesNotExist:
            boat_deckmaterial = None
            render(request, "boater/searchboats.html",
                {"boats": [],
                 "boatsets": [],
                 "boatmodels": boat_models,
                 "boattypes": boat_types,
                 "boatlocations": boat_locations,
                 "boatmaterials": boat_materials,
                 "manufacturers": boat_manufacturers
                })
            boats = boats.filter(deckmaterial =
boat_deckmaterial)
    boat_showernumber = request.GET.get('startshowers')
    if boat_showernumber:
        boat_showernumber = int(boat_showernumber)
        boats = boats.filter(showernumber__gte =
boat_showernumber)
    boat_showernumber = request.GET.get('endshowers')
    if boat_showernumber:
        boat_showernumber = int(boat_showernumber)
        boats = boats.filter(showernumber__lte =
boat_showernumber)
    boat_toiletnumber = request.GET.get('starttoilets')
    if boat_toiletnumber:
        boat_toiletnumber = int(boat_toiletnumber)
        boats = boats.filter(toiletnumber__gte =
boat_toiletnumber)
    boat_toiletnumber = request.GET.get('endtoilets')
    if boat_toiletnumber:
        boat_toiletnumber = int(boat_toiletnumber)
        boats = boats.filter(toiletnumber__lte =
boat_toiletnumber)
    boat_lifejackets = request.GET.get('startlifejackets')
    if boat_lifejackets:
        boat_lifejackets = int(boat_lifejackets)
        boats = boats.filter(lifejackets__gte =
boat_lifejackets)
    boat_lifejackets = request.GET.get('endlifejackets')
    if boat_lifejackets:

```

```

        boat_lif jackets = int(boat_lif jackets)
        boats = boats.filter(lif jackets__lte =
boat_lif jackets)
        boat_navigation = request.GET.get('navigation')
        if boat_navigation:
            boats = boats.filter(navigation = boat_navigation)
        boat_radiocommunication = request.GET.get('radio')
        if boat_radiocommunication:
            boats = boats.filter(radiocommunication =
boat_radiocommunication)
        boat_data["boattypes"] = boat_types
        boat_title = request.GET.get('adtitle')
        if boat_title:
            boats = boats.filter(adtitle__icontains =
boat_title)
        boat_builtyear = request.GET.get('startbuiltyear')
        if boat_builtyear:
            boat_builtyear = int(boat_builtyear)
            boats = boats.filter(builtyear__gte =
boat_builtyear)
        boat_builtyear = request.GET.get('endbuiltyear')
        if boat_builtyear:
            boat_builtyear = int(boat_builtyear)
            boats = boats.filter(builtyear__lte =
boat_builtyear)
        boat_width = request.GET.get('startboatwidth')
        if boat_width:
            boat_width = int(boat_width)
            boats = boats.filter(width__gte = boat_width)
        boat_width = request.GET.get('endboatwidth')
        if boat_width:
            boat_width = int(boat_width)
            boats = boats.filter(width__lte = boat_width)
        boat_length = request.GET.get('startboatlength')
        if boat_length:
            boat_length = int(boat_length)
            boats = boats.filter(length__gte = boat_length)
        boat_length = request.GET.get('endboatlength')
        if boat_length:
            boat_length = int(boat_length)
            boats = boats.filter(length__lte = boat_length)
        boat_conditiontype = request.GET.get('condtype')
        if boat_conditiontype:
            boats = boats.filter(conditiontype =
boat_conditiontype)
        boat_price = request.GET.get('startprice')
        if boat_price:
            boat_price = float(boat_price)
            boats = boats.filter(price__gte = boat_price)
        boat_price = request.GET.get('endprice')
        if boat_price:
            boat_price = float(boat_price)
            boats = boats.filter(price__lte = boat_price)

```

```

boat_hulltype = request.GET.get('hulltype')
if boat_hulltype:
    boats = boats.filter(hulltype = boat_hulltype)
boat_watercapacity = request.GET.get('startcapacity')
if boat_watercapacity:
    boat_watercapacity = int(boat_watercapacity)
    boats = boats.filter(watercapacity__gte =
boat_price)
    boat_watercapacity = request.GET.get('endcapacity')
    if boat_watercapacity:
        boat_watercapacity = int(boat_watercapacity)
        boats = boats.filter(watercapacity__lte =
boat_price)
        boat_enginefuelcapacity =
request.GET.get('startenginefuelcapac')
        if boat_enginefuelcapacity:
            boat_enginefuelcapacity =
int(boat_enginefuelcapacity)
            boats = boats.filter(enginefuelcapacity__gte =
boat_enginefuelcapacity)
            boat_enginefuelcapacity =
request.GET.get('endenginefuelcapac')
            if boat_enginefuelcapacity:
                boat_enginefuelcapacity =
int(boat_enginefuelcapacity)
                boats = boats.filter(enginefuelcapacity__lte =
boat_enginefuelcapacity)
            boat_enginehours = request.GET.get('startenginehours')
            if boat_enginehours:
                boat_enginehours = float(boat_enginehours)
                boats = boats.filter(enginehours__gte =
boat_enginehours)
            boat_enginehours = request.GET.get('endenginehours')
            if boat_enginehours:
                boat_enginehours = float(boat_enginehours)
                boats = boats.filter(enginehours__lte =
boat_enginehours)
            boat_enginehorsepowers =
request.GET.get('starthorsepowers')
            if boat_enginehorsepowers:
                boat_enginehorsepowers =
float(boat_enginehorsepowers)
                boats = boats.filter(enginehorsepowers__gte =
boat_enginehorsepowers)
            boat_enginehorsepowers =
request.GET.get('endhorsepowers')
            if boat_enginehorsepowers:
                boat_enginehorsepowers =
float(boat_enginehorsepowers)
                boats = boats.filter(enginehorsepowers__lte =
boat_enginehorsepowers)
            boat_enginereleaseyear =
request.GET.get('starreleaseyear')

```

```

        if boat_enginereleaseyear:
            boat_enginereleaseyear =
int(boat_enginereleaseyear)
            boats = boats.filter(enginereleaseyear__gte =
boat_enginereleaseyear)
            boat_enginereleaseyear =
request.GET.get('endreleaseyear')
            if boat_enginereleaseyear:
                boat_enginereleaseyear =
int(boat_enginereleaseyear)
                boats = boats.filter(enginereleaseyear__lte =
boat_enginereleaseyear)
            boat_displacement = request.GET.get('displacement')
            if boat_displacement:
                boat_displacement = int(boat_displacement)
                boats = boats.filter(displacement__lte =
boat_displacement)
            boat_enginemade = request.GET.get('enginemade')
            if boat_enginemade:
                boats = boats.filter(enginemade = boat_enginemade)
            boat_enginefueltype = request.GET.get('enginefueltype')
            if boat_enginefueltype:
                boats = boats.filter(enginefueltype =
boat_enginefueltype)
            boats = boats.filter(state = 1)
            from django.db.models import Count
            boats = boats.annotate(userviews_count =
Count("userviews"))
            boats = boats.annotate(userlikes_count =
Count("userlikes"))
            if sortorder == 0:
                boats = boats.order_by("-creationdate")
            elif sortorder == 1:
                boats = boats.order_by("creationdate")
            elif sortorder == 2:
                boats = boats.order_by("userviews_count")
            elif sortorder == 3:
                boats = boats.order_by("-userviews_count")
            elif sortorder == 4:
                boats = boats.order_by("userlikes_count")
            elif sortorder == 5:
                boats = boats.order_by("-userlikes_count")
            elif sortorder == 6:
                boats = boats.order_by("adtitle")
            elif sortorder == 7:
                boats = boats.order_by("-adtitle")
            boatsets = []
            userprofile = request.user
            userprofile = Userprof.objects.get(user = userprofile)
            for boat in boats:
                if userprofile in boat.userlikes:
                    boat.wasliked = True
                else:

```

```

        boat.wasliked = False
        if userprofile in boat.userviews:
            boat.wasviewed = True
        else:
            boat.wasviewed = False
    boatsets =
Userprofboatset.objects.prefetch_related("boats").all()
    boatsets = boatsets.filter(userprof = userprofile)
    for boat in boats:
        boat.boatsets = []
        for boatset in boatsets:
            if boat in boatset.boats:
                boat.boatsets.append(boatset)
            else:
                boatset.boats = []
    boat_data["boats"] = boats
    boat_data["boatsets"] = boatsets
    boat_data["boattypes"] = boat_types
    boat_data["boatmaterials"] = boat_materials,
    return render(request, "boater/searchboats.html",
boat_data)

```

```

from .forms import BoatForm
from .predictmodel import getpricekozmodel

```

```

def boat_ad_publish_view(request):
    if not (request.user.is_authenticated and
request.user.is_active):
        return render(request, "boater/publish.html",
{"form": None })
    try:
        userprofs =
Userprof.objects.select_related("user").all()
        userprof = userprofs.get(user = request.user)
    except Userprof.DoesNotExist:
        boatform = None
        return render(request, "boater/publish.html",
{"form": boatform })
    if request.method == "POST":
        boatform = BoatForm(request.POST)
        if boatform.is_valid():
            boat_ad = boatform.save(commit = False)
            boat_ad.dealerprof = userprof
            boat_ad.pricerecom = getpricekozmodel(boat_ad)
            boat_ad.save()
            return redirect('boatdetailview', boatId =
boat_ad.id)
        else:
            boatform = BoatForm()
            return render(request, "boater/publish.html", {"form":
boatform })

```

```

def boat_ad_change_view(request, boatId):
    if request.user.is_authenticated and
request.user.is_active:
        try:
            userprofs =
Userprof.objects.select_related("user").all()
            userprof = userprofs.get(user = request.user)
        except Userprof.DoesNotExist:
            userprof = None
            return render(request, "boater/edit.html",
{"form": None })
        try:
            boat = Boat.objects.get(id = boatId)
        except Boat.DoesNotExist:
            boat = None
        if boat:
            boatform = BoatForm (request.POST or None,
instance = boat)
            if request.method == "POST" and
boatform.is_valid():
                boat.pricerecom = getpricekozymodel(boat)
                boatform.save()
                return redirect('boatdetailview', boatId =
boatId)
            else:
                boatform = None
            return render(request, "boater/edit.html", {"form":
boatform })

```

```

from .forms import UserprofboatsetForm

```

```

def userprofboatset_add_view(request):
    if not (request.user.is_authenticated and
request.user.is_active):
        return render(request, "boater/addboatset.html",
{"form": None })
    try:
        userprofs =
Userprof.objects.select_related("user").all()
        userprof = userprofs.get(user = request.user)
    except Userprof.DoesNotExist:
        userprof_boatset_form = None
        return render(request, "boater/addboatset.html",
{"form": userprof_boatset_form })
    if request.method == "POST":
        userprof_boatset_form =
UserprofboatsetForm(request.POST)
        if userprof_boatset_form.is_valid():
            userprofboatset =
userprof_boatset_form.save(commit = False)
            userprofboatset.userprof = userprof

```

```

        userprofboatset.save()
        return redirect('boatsetboatsview', boatsetId =
userprofboatset.id)
    else:
        userprof_boatset_form = UserprofboatsetForm()
        return render(request, "boater/addboatset.html",
{"form": userprof_boatset_form })

def userprofboatset_change_view(request,
userprofboatsetId):
    if request.user.is_authenticated and
request.user.is_active:
        try:
            userprofs =
Userprof.objects.select_related("user").all()
            userprof = userprofs.get(user = request.user)
        except Userprof.DoesNotExist:
            userprof = None
            return render(request, "boater/editset.html",
{"form": None })
        try:
            userprof_boatset =
Userprofboatset.objects.select_related("userprof",
"userprof__user").get(id = userprofboatsetId)
        except Userprofboatset.DoesNotExist:
            userprof_boatset = None
            if not (userprof_boatset and
userprof_boatset.userprof == userprof):
                return render(request, "boater/editset.html",
{"form": None })
            if userprof_boatset:
                userprof_boatset_form =
UserprofboatsetForm(request.POST or None, instance =
userprof_boatset)
                if request.method == "POST" and
userprof_boatset_form.is_valid():
                    userprof_boatset_form.save()
                    return redirect('boatsetboatsview', boatsetId =
userprofboatsetId)
            else:
                userprof_boatset_form = None
                return render(request, "boater/editset.html", {"form":
userprof_boatset_form })

```

```

from .forms import BoatsaleForm

```

```

def boatsale_add_view(request, boatId):
    try:
        boats = Boat.objects.all()
        boat = boats.get(id = boatId)

```

```

        except Boat.DoesNotExist:
            boat = None
            return render(request, "boater/addboatsale.html",
{"boat": None })
            if not (request.user.is_authenticated and
request.user.is_active):
                return render(request, "boater/addboatsale.html",
{"form": None })
            try:
                userprofs =
Userprof.objects.select_related("user").all()
                userprof = userprofs.get(user = request.user)
            except Userprof.DoesNotExist:
                boatsale_form = None
                return render(request, "boater/addboatsale.html",
{"form": boatsale_form })
            if request.method == "POST":
                boatsale_form = BoatsaleForm(request.POST)
                if boatsale_form.is_valid():
                    boatsale = boatsale_form.save(commit = False)
                    boatsale.userprof = userprof
                    boatsale.boat = boat
                    boatsale.state = 0
                    boatsale.save()
                    return redirect('ownboatsalesview')
            else:
                boatsale_form = BoatsaleForm()
                return render(request, "boater/addboatsale.html",
{"form": boatsale_form })

def own_userprof_boatsales_view(request, sortorder = 0):
    if not request.user.is_authenticated:
        render(request, "boater/oboatsales.html",
            {"boatsales": None,
             "boatsets": []
            })
    try:
        userprofs =
Userprof.objects.select_related("user").all()
        userprof = userprofs.get(user = request.user)
    except Userprof.DoesNotExist:
        userprof = None
        return render(request, "boater/oboatsales.html",
            {"boatsales": None,
             "boatsets": []
            })
    boatsales = Boatsale.objects.select_related("boat",
        "boat__boatmodel",
        "boat__location",
        "boat__boatmodel__boatmanufacturer",
        "boat__boatmodel__boattype").prefetch_related("boat__userviews",

```

```

        "boat__userlikes").all()
boatsales = boatsales.filter(userprof = userprof)
from django.db.models import Count
if sortorder == 0:
    boatsales = boatsales.order_by("-apllicationdate")
elif sortorder == 1:
    boatsales = boatsales.order_by("apllicationdate")
boatsets = []
for boatsale in boatsales:
    if userprofile in boatsale.boat.userlikes:
        boat.wasliked = True
    else:
        boat.wasliked = False
    if userprofile in boatsale.boat.userviews:
        boat.wasviewed = True
    else:
        boat.wasviewed = False
boatsets =
Userprofboatset.objects.prefetch_related("boats").all()
boatsets = boatsets.filter(userprof = userprofile)
for boatsale in boatsales:
    boatsale.boat.boatsets = []
    for boatset in boatsets:
        if boatsale.boat in boatset.boats:
            boatsale.boat.boatsets.append(boatset)
        else:
            boatset.boats = []
return render(request, "boater/boatsales.html",
              {"boatsales": boatsales,
               "boatsets": boatsets
              })

def boat_boatsales_view(request, boatId, sortorder = 0):
    try:
        boats = Boat.objects.all()
        boat = boats.get(id = boatId)
    except Boat.DoesNotExist:
        boat = None
        return render(request, "boater/boatboatsales.html",
{"boat": None,
    "boatsales": boatsales })
    if not request.user.is_authenticated:
        render(request, "boater/boatboatsales.html",
              {"boat": None,
               "boatsales": []
              })
    try:
        userprofs =
Userprof.objects.select_related("user").all()
        userprof = userprofs.get(user = request.user)
    except Userprof.DoesNotExist:
        userprof = None

```


ДОДАТОК В
Слайди презентації

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Дипломна кваліфікаційна робота магістра на тему:

ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
ПІДТРИМКИ ПРОДАЖУ ЧОВНІВ. RESEARCH AND SOFTWARE
IMPLEMENTATION OF METHODS FOR SUPPORTING BOAT SALES

Виконала студентка групи КНТ-123м
Керівник к.т.н., доцент

Дмитро КОЗІЙ
Валерій ЛЬОВКІН

2024

Рисунок В.1 – Початковий слайд

ОБ'ЄКТ, ПРЕДМЕТ, МЕТА РОБОТИ

- Об'єкт роботи – процес продажу човнів.
- Предмет роботи – методи підтримки продажу човнів.
- Мета роботи – розробити програмне забезпечення підтримки продажу човнів для забезпечення рішень стосовно визначення вартості продажу на основі даних.

2

Рисунок В.2 – Об'єкт, предмет, мета роботи

ЗАВДАННЯ РОБОТИ

- виконати аналіз підтримки продажу човнів;
- розробити метод визначення рекомендованої вартості продажу човна;
- виконати проектування програми підтримки продажу човнів;
- реалізувати програму підтримки продажу човнів;
- виконати експериментальне дослідження прогнозування вартості продажу човна.

3

Рисунок В.3 – Завдання роботи

ПОРІВНЯННЯ АНАЛОГІВ РОЗРОБКИ ВЕБЗАСТОСУНКУ ПІДТРИМКИ ПРОДАЖУ ЧОВНІВ

Показник	Boatsales.com.au	AutoRia	Розробка
Багатопараметризований пошук човнів	Розширений	Стандартний	Розширений
Характеристики човна в оголошенні	Розширені	Стандартні	Розширені
Спеціалізація	Човни	Автомобілі, нерухомість, водний транспорт	Човни
Виставлення човна на продаж	+	+	+
Створення добірок човнів	–	–	+
Рекомендування вартості продажу човна для власника	–	–	+

4

Рисунок В.4 – Порівняння аналогів розробки вебзастосунку підтримки продажу човнів

МЕТОД ПРОГНОЗУВАННЯ ВАРТОСТІ ПРОДАЖУ ЧОВНА

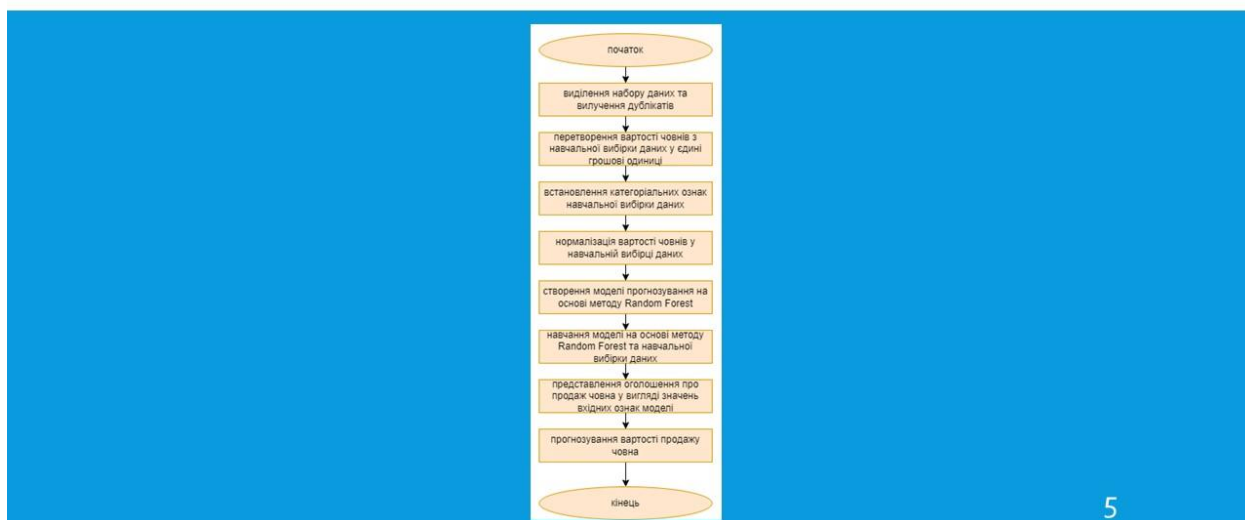


Рисунок В.5 – Метод прогнозування вартості продажу човна

ВИБІР ВЕБФРЕЙМВОРКУ ДЛЯ РОЗРОБКИ ПРОГРАМИ ПІДТРИМКИ ПРОДАЖУ ЧОВНІВ

	Django	Laravel
Тип пропонованого рішення	Вебфреймворк	Вебфреймворк
Мова створення	Python	PHP
Підтримка високого трафіку	Придатний	Потребує додаткової оптимізації для забезпечення того ж рівня
Рівень підтримки машинного навчання	Високий, широка кількість актуальних засобів	Гірший
Безпека	Високий рівень підтримки, велика кількість практик	Високий рівень підтримки

6

Рисунок В.6 – Вибір вебфреймворку для розробки програми підтримки продажу човнів

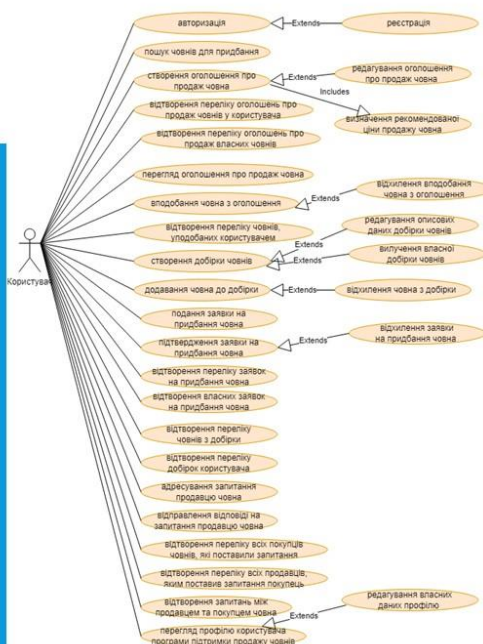


Рисунок В.7 – Діаграма прецедентів ко-ристувача програми під-тримки продажу човнів

СХЕМА БАЗИ ДАНИХ

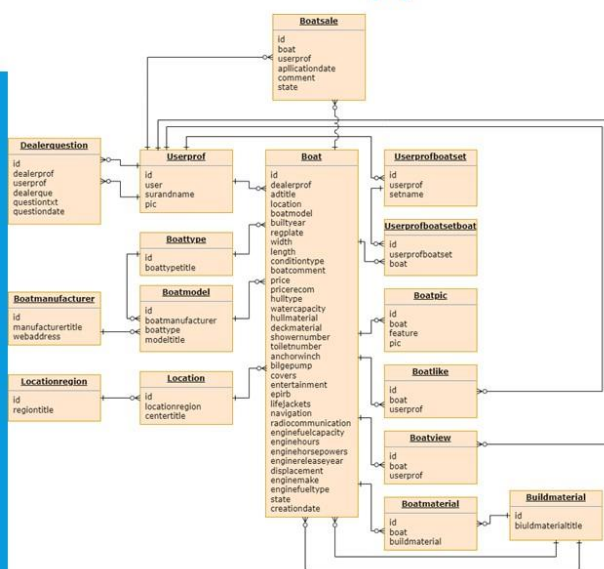


Рисунок В.8 – Схема бази даних

СХЕМА ФУНКЦІОНУВАННЯ ПРОГРАМИ ПІДТРИМКИ ПРОДАЖУ ЧОВНІВ

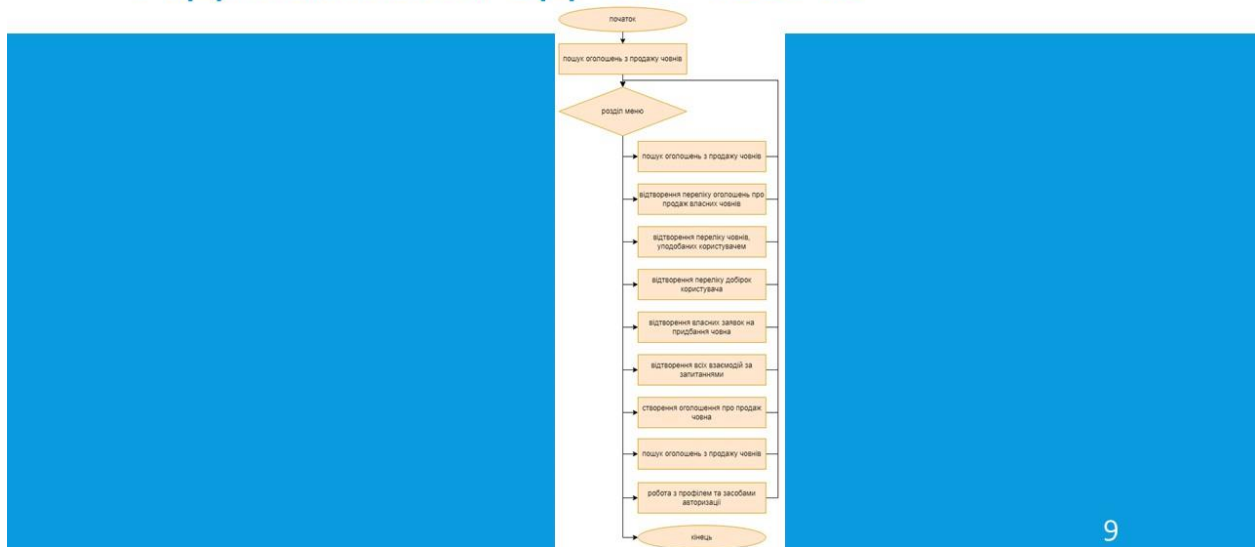


Рисунок В.9 – Схема функціонування програми підтримки продажу човнів

СТОРІНКА ДОБІРКИ ЧОВНІВ

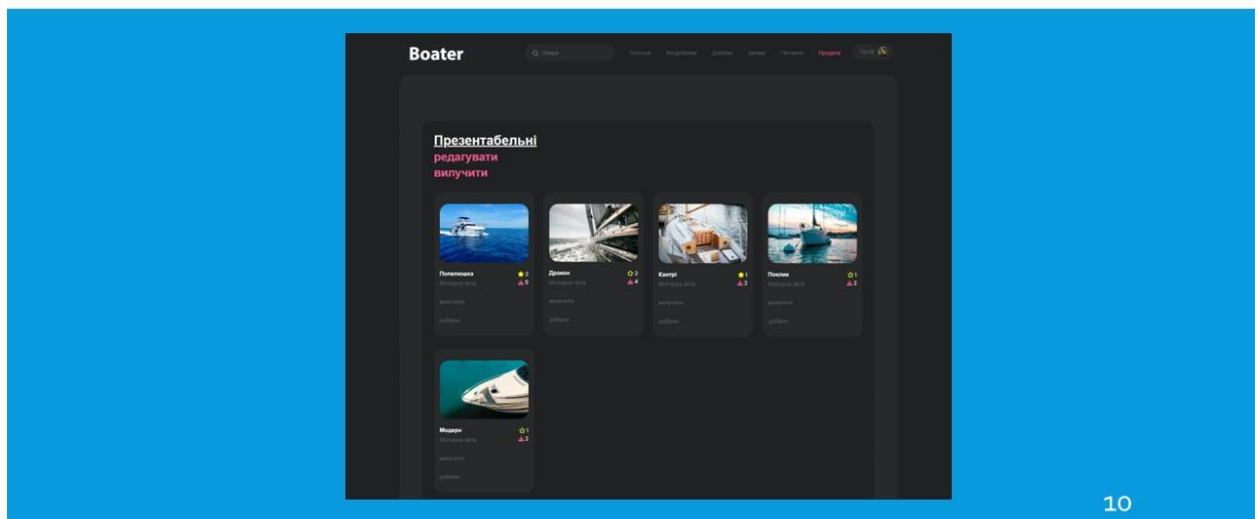


Рисунок В.10 – Сторінка добірки човнів

СТОРІНКА ОГОЛОШЕННЯ ПРО ПРОДАЖ ЧОВНА

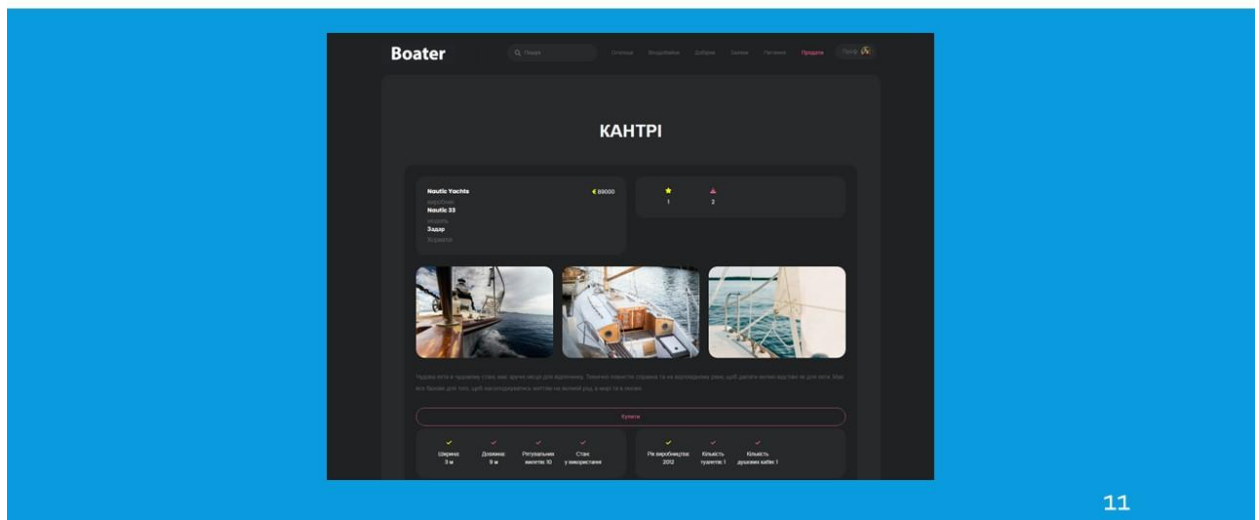


Рисунок В.11 – Сторінка оголошення про продаж човна

ПУБЛІКАЦІЯ ОГОЛОШЕННЯ ПРО ПРОДАЖ ЧОВНА

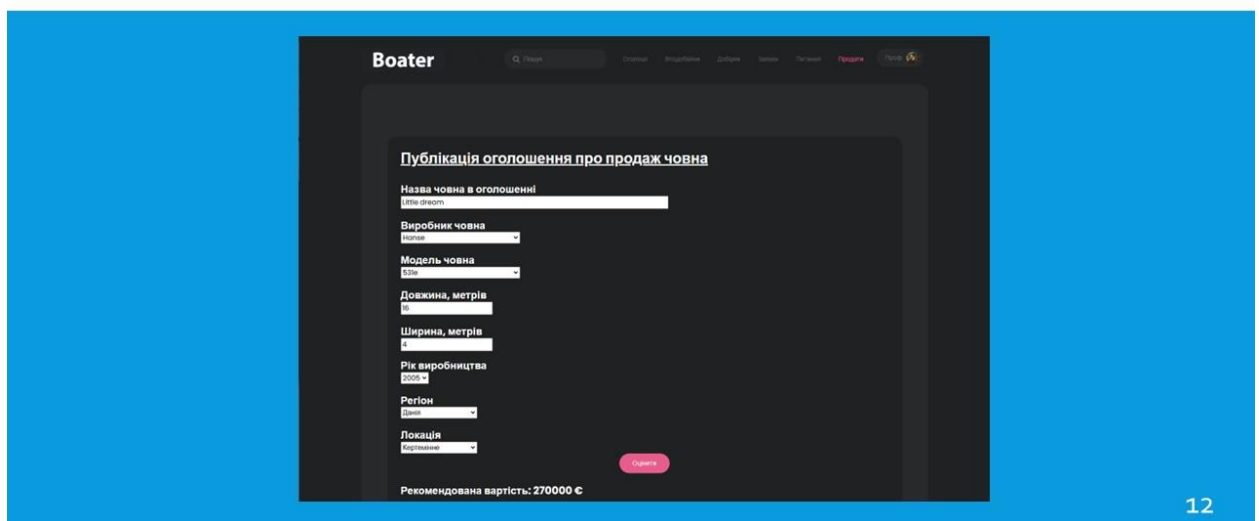


Рисунок В.12 – Публікація оголошення про продаж човна

ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ

Модель	MAE	RMSE	MSE
Дерево рішень (9 ознак)	0,001533	0,010117	0,000102
Дерево рішень (8 ознак)	0,001511	0,010189	0,000104
Дерево рішень (7 ознак)	0,001346	0,00774	5,99122E-05
Дерево рішень (6 ознак)	0,00131	0,007839	6,576724E-05
Дерево рішень (5 ознак)	0,001298	0,008052	6,483627E-05
Дерево рішень (4 ознаки)	0,001305	0,008357	6,984261E-05
Дерево рішень (3 ознаки)	0,001347	0,008527	7,270321E-05
Дерево рішень (2 ознаки)	0,001556	0,008009	6,41441E-05
Метод опорних векторів	0,092738	0,093114	0,00867
Random Forest за запропонованим методом (9 ознак)	0,001141	0,008006	6,409851E-05
Random Forest за запропонованим методом (8 ознак)	0,001141	0,007953	6,324723E-05
Random Forest за запропонованим методом (7 ознак)	0,001127	0,007824	6,121165E-05
Random Forest за запропонованим методом (6 ознак)	0,001119	0,007688	5,911011E-05
Random Forest за запропонованим методом (5 ознак)	0,001075	0,007585	5,752831E-05
Random Forest за запропонованим методом (4 ознаки)	0,001113	0,007574	5,737047E-05
Random Forest за запропонованим методом (3 ознаки)	0,001131	0,007796	6,07748E-05
Random Forest за запропонованим методом (2 ознаки)	0,001491	0,009493	9,01143E-05

13

Рисунок В.13 – Експериментальне дослідження

ВИСНОВКИ

- Створена програма призначена для підтримки виставлення власних нових або вживаних човнів для продажу шляхом створення відповідних оголошень, створення, підтвердження або відхилення заявок на придбання човнів, пошуку необхідних користувача човнів. Користувачам надається можливість створювати власні добірки човнів, щоб повертатись до раніше відібраних.
- Наукова новизна роботи полягає в методі прогнозування вартості продажу човна, який на основі ознак оголошення про продаж човна визначає створення і навчання моделі як ансамблю дерев рішень Random Forest, що дозволяє провести рекомендаційне оцінювання вартості човна для формування підсумкового оголошення.

14

Рисунок В.14 – Висновки